Universidad de las Ciencias Informáticas

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Título: "Sistema de Control de *Hardware* y Medios Básicos del Centro de Identificación y Seguridad Digital".

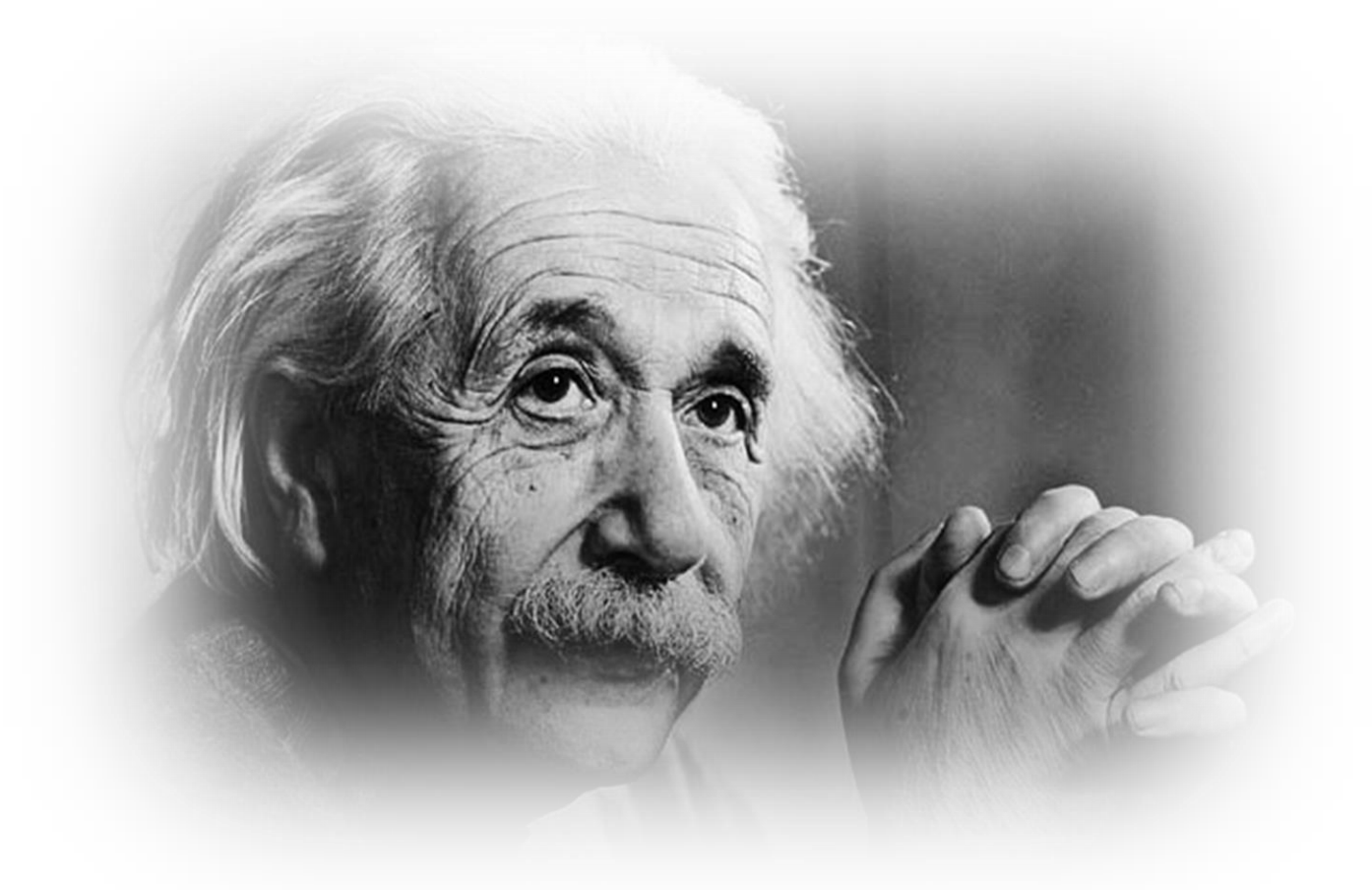
Autores: Arailys Castellano Ferro
Germán Núñez Montero

Tutores: Ing. Ronaldo Castro Milán
Ing. Yayneris Zambrana Hernández

Consultante: Msc. Damaris Cruz Amarán

La Habana, Junio de 2014
"Año 56 de la Revolución"

Pensamiento



Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber.

Albert Einstein

Declaración de autoría

Declaramos ser autores del presente trabajo de diploma titulado “Sistema de Control de *Hardware* y Medios Básicos del Centro de Identificación y Seguridad Digital (CISED)” y reconocemos 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 2014.

Arailys Castellano Ferro

_____

Autores: Arailys Castellano Ferro

Germán Núñez Montero

_____

Autores: Germán Núñez Montero

Yayneris Zambrana Hernández

_____

Tutor: Ing. Yayneris Zambrana Hernández

Ronaldo Castro Milán

_____

Tutor: Ing. Ronaldo Castro Milán

Datos del contacto

Tutor: Ing. Ronaldo Castro Milán.

Especialidad de graduación: Ciencias Informáticas.
Categoría docente: Instructor.
Correo electrónico: rcastro@uci.cu

Tutor: Ing. Yayneris Zambrana Hernández.

Especialidad de graduación: Ciencias Informáticas.
Categoría docente: Instructor.
Correo electrónico: yzambrana@uci.cu

Agradecimientos

Arailys Castellano Ferro

A mi mami Deysi y mi papito Joaquín por estar siempre conmigo en todos los momentos de mi vida, por su apoyo y amor, por dármelo todo en la vida y por confiar en todo momento en que yo lograría estar discutiendo mi tesis el día de hoy.

A mi familia, que de una forma u otra han colaborado porque este momento se hiciera realidad, en especial a mi tío Tony, gracias por confiar en mí.

A mi novio Ariel que ha sido mi punto de apoyo todo este tiempo en la ausencia de mis papis, brindándome su apoyo desmedido y aguantando mis malacrianzas; a su familia, que ya es mi familia, por ser tan cariñosos conmigo… a Niurka, Yusi (Yaimé), Ariel Papá, Tecla, Abu (Ramón), gracias por su preocupación.

A mi compañero de tesis Germán por soportarme todo este tiempo que hemos pasado juntos, muchas gracias!

A mis SÚPER TURORES Yayneris y Ronaldo (el figurín), por ser personas incondicionales y estar ahí cuando más los he necesitado.

A mis amistades, esos que han significado mucho en mi vida por brindarme su amistad y estar presentes cuando las necesité: **Liester**, Eliena, Lianet, Dennis, a los nenes del grupo, a todos los que me quieren y a los que no también… GRACIAS.

Germán Núñez Montero

A mi mamá Clara por siempre estar ahí para mí, por ser padre y madre para mí, por sobre todas las cosas educarme por el camino del bien, y enseñarme que la vida es de sacrificios.

A mi madrina y a mi padrino por apoyarme y darme fuerzas, por hacerme ver la vida desde un punto de vista diferente.

A mi hermana por tanta lucha y tantas peleas. A mi hermanos Pedrito, Alexander y Yordys.

A mis tíos Irene y Leonel por ser una guía para mí, y por sobre todas las cosas hacer que yo me esforzara por no desilusionarlos.

A mis primas María de los Ángeles y Yipsy por acogerme, apoyarme, regañarme y sobre todo hacer que me esforzara. A mi primo Maikel, por esforzarse y permitirme ser un ejemplo para él, su madre Vilma, a Juanita a ustedes gracias también.

A Ramón por hacer que cada día yo mejorara más, porque siempre estar criticándome y nunca decirme que lo que hacía estaba bien.

A mi novia Orleanna por quererme y aunque está lejos en estos momentos, siempre me ha apoyado.

A mi amiga inseparable Leanni, por ser una hermana para mí aquí.

A una persona que entró hace poco en mi vida pero marcó bien duro en ella (YRA), gracias por todo.

A Thais porque fue una parte importante de mi vida, a su familia por quererme como un miembro más de ella.

A mi compañera de tesis Arailys por tantos días de desvelo y por tanto sacrificio, y al final cuando todo parecía derrumbarse siempre estuvo ahí apoyándome.

A mis queridos tutores que siempre nos defendieron y fueron hasta el último momento con nosotros, a ellos gracias por su guía.

A el Combo y sus sobrevivientes.

A Lianet, Mily, Eliena, Alberto, Robín, Adriancito (el puro), Yasiel, Adolfo, Miriela, a todos ellos gracias por ser parte del grupo.

A Pepe por apoyarme y por el transporte siempre que pudo estuvo disponible.

A la gente del barrio, a los que creyeron y los que no creyeron, a todos ustedes gracias.

Dedicatoria

Arailys Castellano Ferro

Dedico mi Trabajo de Diploma a mi mami, por haberme apoyado en todo momento de mi vida y de la carrera, te amo mami.

A mis padres (Deysi y Joaquín), quienes estarán orgullosos de verme al fin, graduada como ingeniera.

A toda mi familia: mis abuelos (Mima y Papi), tías, tíos, primas, primos, que siempre me han apoyado en aras de cumplir mis sueños.

A todos muchas gracias!!!!!

Germán Núñez Montero

Le dedico mi trabajo de diploma a mi abuela Teresa que ya no está presente y no pudo ver a su nieto graduado de universitario.

Se lo dedico a mi mamá por siempre apoyarme y por hacer de madre y padre para mí, por quitarse hasta lo que no tenía para dármelo.

A mi Hermana Lucrecia y mi sobrino Cristhian que son una de las cosas más grandes que yo tengo en la vida.

A mis padrinos Nancy y Jorge, por siempre apoyarme.

A quien siempre me apoyó y que siempre estuvo pendiente de mis resultados a mi tía Irene, mi tío Leonel, mis primas María de los Ángeles y Yipsy.

En fin, se lo dedico a todas aquellas personas que siempre confiaron en mí.

Resumen

En el presente trabajo se describe el proceso de control de *hardware* y medios básicos que se desarrolla actualmente en el Centro de Identificación y Seguridad Digital (CISED). Este proceso se realiza de manera engorrosa, debido a que depende de una serie de pasos a seguir y planillas que deben ser llenadas manualmente y a diario por un grupo de personas designadas. Con el propósito de tener un control centralizado e informatizar este proceso se define como objetivo general de este trabajo: Desarrollar una aplicación web que perfeccione el control de *hardware* y medios básicos que se realiza en el Centro de Identificación y Seguridad Digital.

Para el cumplimiento de este objetivo se realiza la investigación de algunos de los sistemas similares al que se desea desarrollar, así como el análisis y definición de la metodología de desarrollo a seguir, las tecnologías, herramientas y lenguajes de programación necesarios para la implementación de dicho sistema.

Finalmente se muestra los resultados de las pruebas funcionales realizadas para comprobar la calidad del Sistema de Control de *Hardware* y Medios Básicos del Centro de Identificación y Seguridad Digital; el cual representa una mejora total de este proceso permitiendo la correcta asignación y planificación de los activos del centro.

Palabras Clave: activos fijos tangibles, control de inventario, medios básicos.

Índice General

Índice de Figuras

Índice de Tablas

INTRODUCCIÓN

El sistema empresarial en la actualidad constituye un elemento de gran importancia para el desarrollo económico de cada país. Las empresas en el mundo moderno viven un escenario económico que impone a los directivos una reflexión en los mecanismos de la gestión empresarial ante la turbulencia e incertidumbre de su entorno. Es decir, hay que lograr producciones eficientes y competitivas para lograr el éxito de la empresa.

El sistema administrativo representa el significado general de toda organización, teniendo en cuenta que al construir una empresa se debe estudiar y planificar las actividades que se realizan en la actualidad, con una visión futura para un mejor control en el funcionamiento de la organización; y con el fin de obtener una buena dirección, coordinación y control de las actividades laborales de la misma (1).

En particular, el desarrollo de la empresa cubana tiene nuevos retos en la gestión de la actividad económica que realiza; elemento de gran importancia en el perfeccionamiento empresarial como vía para situar a las empresas en condiciones imprescindibles de la efectividad en el uso de los recursos. Se debe ser capaz de proyectar su desarrollo con visión de futuro, planificar cada etapa de actividad donde se ha de prever la disponibilidad de los recursos de todo tipo que hoy en día son costosos y escasos en el país.

Los recursos materiales o activos fijos tangibles como también se les puede llamar en este contexto, constituyen los bienes materiales que una entidad utiliza de manera continua en sus operaciones. Los mismos requieren de un control a partir de un sistema de modelos que reflejan su destino y uso; por lo que es necesario realizarlo de forma detallada. Esto permite mantener un control estricto de los mismos y servir como base para el chequeo físico de los bienes.

La Universidad de las Ciencias Informáticas (UCI) es una institución docente que tiene como propósito formar profesionales informáticos altamente calificados, producir *software* y servicios informáticos a partir de la vinculación estudio-trabajo como modelo de formación. Por lo que cuenta con varios centros de producción de *software*, entre los que se encuentra el Centro de Identificación y Seguridad Digital (CISED); el cual no queda exento de las nuevas tendencias empresariales que se vienen desarrollando en el país a raíz del perfeccionamiento empresarial.

Este centro tiene bajo su responsabilidad una serie de recursos materiales o activos fijos tangibles (*hardware* y medios básicos) los cuales se controlan, algunos cada cierto período de tiempo y otros diariamente dependiendo del tipo de activo que sea; haciendo variar también la manera de realizar el control.

Para el control de *hardware* se utiliza *OCS Inventory*, herramienta informática que ofrece la información de cada uno de los componentes que constituyen los ordenadores del centro; además cada día en la entrega de guardia de los técnicos trabajadores del centro, estos son responsables de revisar cada uno de los ordenadores en cada laboratorio del centro; y una vez finalizado este proceso, deben llenar una planilla definida que recoge la información necesaria, así como las incidencias encontradas en el momento del control de estos recursos. Estas planillas se le entregan al Responsable de Tecnología del centro, el cual recopila los datos; y llena un informe que debe enviar a la facultad.

Por otro lado, el control de los medios básicos se realiza de manera engorrosa, el económico del centro es el responsable de ir a cada local y controlar que coincidan los datos de los medios existentes con los datos de las planillas guardadas en el último control de inventario realizado. Esta persona es la encargada de llenar varias planillas definidas para la recopilación de los datos, y enviar reportes a la facultad cada cierto tiempo; además de tener el control total y actualizado de cada medio básico del centro.

La información que se genera en cada momento que se realiza alguna actividad de control se registra de manera manual, lo que provoca acumulación de cierta cantidad de información en formato duro que hace difícil poder comparar la entrega actual de cierta información con alguna realizada en otro momento; o sea, es imposible contar con una información actualizada en tiempo real de los activos del centro. Además, estas planillas u hojas de inventario son siempre vulnerables a pérdidas o modificaciones.

Por otro lado, el registro de los movimientos de medios básicos se realiza de manera manual; corriendo el riesgo de que existan movimientos no registrados en la hoja de inventario y no siempre se cuente con la información más actualizada. De manera general es engorroso consultar información o determinados datos, dado que esta se encuentra en libros Excel, donde no se permite visualizar reportes específicos con la información requerida.

Como se puede apreciar, el proceso de control de *hardware* y medios básicos se realiza de manera independiente, se trabaja por separado y en muchas ocasiones no es posible darse cuenta de los errores que se han cometido, debido a que la información no se tiene almacenada de forma centralizada, lo que implica un control deficiente de los activos fijos del centro.

Después de haber analizado la **situación problemática** anteriormente expuesta, se plantea como **problema a resolver** ¿Cómo perfeccionar el proceso de control de *hardware* y medios básicos del Centro de Identificación y Seguridad Digital?

Partiendo del problema planteado se tiene como **objeto de estudio** el proceso de control de inventario.

Para dar solución al problema planteado se traza como **objetivo general**: Desarrollar una aplicación web que perfeccione el control de *hardware* y medios básicos del Centro de Identificación y Seguridad Digital.

Siendo el **campo de acción** el proceso de control de *hardware* y medios básicos en el Centro de Identificación y Seguridad Digital.

El objetivo general se divide en los siguientes **objetivos específicos:**

- Analizar el estado del arte del tema de investigación.
- Diseñar el sistema de control de *hardware* y medios básicos.
- Implementar el sistema de control de *hardware* y medios básicos.
- Realizar pruebas funcionales al sistema desarrollado.

Para dar solución a los objetivos planteados anteriormente se proponen las siguientes **Tareas de investigación:**

1. Análisis de la bibliografía referente al tema de investigación.
2. Elaboración del marco teórico de la investigación.
3. Levantamiento de los requisitos funcionales y no funcionales del sistema a desarrollar.
4. Definición de la arquitectura del sistema de control de *hardware* y medios básicos.
5. Definición del modelo de datos del sistema.
6. Definición de los patrones de diseño del sistema.
7. Definición de los estándares de codificación del sistema.
8. Implementación del sistema de control de *hardware* y medios básicos.
9. Elaboración del diagrama de despliegue del sistema implementado.
10. Diseño de los casos de pruebas a realizar en el sistema implementado.
11. Realización de las pruebas funcionales al sistema implementado.
12. Elaboración del documento de investigación.

Durante el desarrollo de la investigación se utilizaron los métodos científicos que a continuación se detallan:

Métodos teóricos:

- El método **modelación**: se utiliza para representar los diagramas generados en el diseño, los cuales facilitan un mejor entendimiento del sistema.

- El método **analítico-sintético**: se utiliza para sintetizar los elementos más importantes de las fuentes bibliográficas analizadas, y de esta manera poder identificar las tecnologías y herramientas a usar para el desarrollo del sistema.

Métodos empíricos:

- El método **entrevista**: se utiliza para obtener y recopilar toda la información necesaria para el diseño e implementación del sistema de control de *hardware* y medios básicos del CISED.
- El método **observación** se utiliza para observar cómo se realiza actualmente el control de *hardware* y medios básicos en el centro productivo.

Resultados esperados:

Se obtendrá un sistema informático para el control de *hardware* y medios básicos del CISED; que permitirá a los directivos realizar la gestión de los recursos materiales del centro y mantener toda la información centralizada para un control estricto de los mismos.

El presente trabajo se estructura de la siguiente manera:

Capítulo 1 Fundamentación Teórica: Se realiza el estudio del estado del arte referente al tema de investigación. Se describen los conceptos fundamentales asociados al dominio del problema y al objeto de estudio, así como un análisis de la situación actual existente. También se describen y definen las herramientas, tecnologías y lenguajes a ser usados para el desarrollo de la solución.

Capítulo 2 Características del Sistema: Después de realizada la fundamentación teórica de la investigación, en este capítulo se presenta una descripción de la propuesta de solución, que siguiendo la metodología definida para su desarrollo permite realizar la planificación por etapas, la especificación de sus funcionalidades y el diseño del sistema mediante el modelado de los diagramas del diseño.

Capítulo 3 Implementación y Pruebas: En este capítulo se presentan los estándares de codificación utilizados, se muestra la distribución física del sistema entre los diferentes nodos a través del diagrama de despliegue y por último, se presentan los resultados de las principales pruebas funcionales realizadas al sistema para verificar su correcto funcionamiento.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Se realiza el estudio del estado del arte referente al tema de investigación. Se describen los conceptos fundamentales asociados al dominio del problema y al objeto de estudio, así como un análisis de la situación actual existente. También se describen y definen las herramientas, tecnologías y lenguajes a ser usados para el desarrollo de la solución.

1.1. Conceptos asociados al dominio del problema

Inventario

Registro documental de los bienes y demás objetos pertenecientes a una persona o entidad, que se encuentra realizado a partir de mucha precisión en la plasmación de los datos (2).

Sistema de control de inventario

Verificación periódica de las existencias de materiales, equipo, muebles e inmuebles con que cuenta una dependencia o entidad, a efecto de comprobar el grado de eficacia en los sistemas de control administrativo, el manejo de los materiales y el método de almacenaje (3).

Activos Fijos Tangibles

Los Activos Fijos Tangibles constituyen los bienes materiales que una entidad utiliza de manera continua en sus operaciones, con un carácter permanente, los cuales sufren una depreciación durante su vida útil que afectará al final el nivel de utilidades (4).

1.2. Análisis de herramientas para el control de inventarios

Como parte de la investigación se realiza un análisis de varias herramientas similares a la que se desea desarrollar para dar solución a la problemática planteada. A continuación se muestra una descripción de las mismas:

1.2.1. Inventario Abierto de Software y Computadoras (OCS Inventory)

OCS Inventory (*Open Computer and Software Inventory*) es una herramienta que permite realizar un inventario diario de todos los equipos de una red. Utiliza un agente, el cual realiza los inventarios en los clientes y un servidor de administración, que centraliza los resultados del mismo permitiendo visualizarlos. OCS Inventory es liberado bajo la licencia GNU *General Public License* (Licencia Pública General),

versión 2.0 (GNU GPLv2). No permite realizar tareas programadas como pudiera ser la realización de un inventario a una hora especificada (5).

1.2.2. Administración Dinámica de Red (NetSupport DNA)

NetSupport DNA (*Dynamic Network Administration*) es una completa solución modular que ofrece inventario de *software* y *hardware* automatizado, control de usuarios, gestión de licencias, distribución de *software*, alertas e informes. Identifica las aplicaciones de *software* que no se usan. Además elimina posibles infecciones por virus al controlar el uso de USB, CD, DVD (6).

1.2.3. Rodas XXI

Rodas XXI es un sistema creado por CITMATEL[1] para automatizar el funcionamiento de cualquier empresa o unidad presupuestada. El sistema cuenta actualmente con ocho módulos:

- Finanzas
- Contabilidad
- **Activos Fijos**
- Nóminas
- **Inventario**
- Facturación
- Recursos Humanos
- Telecobranzas

Inventario

El módulo de Inventario permite tener un control detallado de los inventarios de la entidad, realizando su contabilización en el mismo momento que se registra un movimiento. Se realizan los cierres diariamente cada vez que se realicen movimientos.

Este módulo permite además, visualizar información correspondiente a períodos anteriores, tan sólo con cambiar de período contable a otros anteriores ya cerrados, aunque en dichos períodos no podrá realizar ninguna operación.

Activos Fijos

[1] CITMATEL: Empresa de Tecnología y Servicios Telemáticos Avanzados.

El módulo de **Activos Fijos** permite tener un control detallado de los activos fijos de la entidad, realizando su contabilización en el mismo momento que se registra un movimiento. Permite el control por separado de los activos fijos que se encuentran en el almacén de los que se encuentran en explotación (7).

1.2.4. Configurador Automático y Colector de Informaciones Computacionales (CACIC)

CACIC es un sistema de inventario de *hardware* y *software*, que se basa en un agente capaz de obtener un diagnóstico completo sobre los cambios producidos en componentes de *hardware* y proporciona la información a una estación o servidor. Además su código fuente es libre.

El sistema tiene varias características entre las que destacan:

- Soporte de inventario a la plataforma Windows 32 bits y Linux.
- Captura automatizada de los datos de *Hardware* y *Software*.
- Soporte a múltiples idiomas (8).

Después de realizado el estudio de estas herramientas se tiene la siguiente valoración:

- ***OCS Inventory:*** es un sistema que se utiliza actualmente en el CISED, pero presenta como desventaja para su utilización, que duplica la información de cualquier cambio que se realice en el ordenador. Tampoco permite llevar un control del historial de acciones realizadas con anterioridad.
- ***NetSupport DNA*****:** se descartó su utilización ya que presenta como desventaja que solo se notifica en la consola de administración web la ocurrencia de un cambio en el *hardware* y no hay forma de conocer si fue modificado un inventario en la caché[2] del agente cliente.
- **Rodas XXI:** se caracteriza por abordar solamente partes del problema de la gestión de la empresa o la unidad presupuestada, no soporta mecanismos estándares de integración con otras aplicaciones donde la mayoría fueron desarrollados para un ambiente multiusuario, casi ninguno bajo conceptos de informática multicapa y distribuida en la red (9).
- **CACIC:** presenta como inconveniente para ser utilizada, que la única vía de notificar cambios en el *hardware* es mediante correo y no hay forma de conocer si fue modificado un inventario en la caché del agente cliente.

Finalmente, se determina que es necesario desarrollar un sistema que permita el control de *hardware* y medios básicos en el CISED, de manera que cumpla con los requisitos especificados por el cliente.

[2] Caché: Memoria de sitios a los que se accede con frecuencia, el cual es usado por la unidad central de procesamiento (CPU) para reducir el tiempo de acceso a datos ubicados en la memoria principal que se utilizan con más frecuencia.

1.3. Ambiente de desarrollo

Debido a las líneas de investigación y desarrollo que se tienen definidas en el CISED; como son la Biometría, Seguridad Digital y aplicaciones para Tarjetas Inteligentes ha sido importante la aplicación de metodologías ágiles y tecnologías .NET como alternativa para el desarrollo de productos estables y maduros. En la presente investigación se define .NET como la tecnología clave para el desarrollo de la propuesta de solución aprovechando así las políticas y experiencias del CISED en las distintas tecnologías involucradas.

1.3.1. Metodología de desarrollo de software. Desarrollo Basado en Funcionalidades (FDD)

Una metodología es un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo *software* (10). Las mismas tienen como objetivo estructurar, planificar y controlar dicho proceso, por lo que indican paso a paso todas las actividades que se deben efectuar para alcanzar el producto deseado, indicando además quién debe participar en el desarrollo de cada una de las actividades y qué papel deben tener en las mismas.

Existen dos tipos de clasificaciones de metodologías, las metodologías tradicionales y las ágiles. Estas últimas proporcionan una serie de pautas y principios junto a técnicas pragmáticas que harán la entrega del proyecto menos complicada y más satisfactoria tanto para los clientes como para los equipos de entrega.

Entre estas metodologías se encuentra la metodología Desarrollo Basado en Funcionalidades (*Feature Driven Development,* FDD) es un enfoque ágil para el desarrollo de sistemas. Dicho enfoque no hace énfasis en la obtención de los requisitos, sino en cómo se realizan las fases de diseño y construcción. Las iteraciones se deciden teniendo en cuenta las funcionalidades, que representan fragmentos del *software* con significado para el cliente (11).

La misma presenta cinco fases secuenciales que son mostradas a continuación (Ver **Figura 1**):

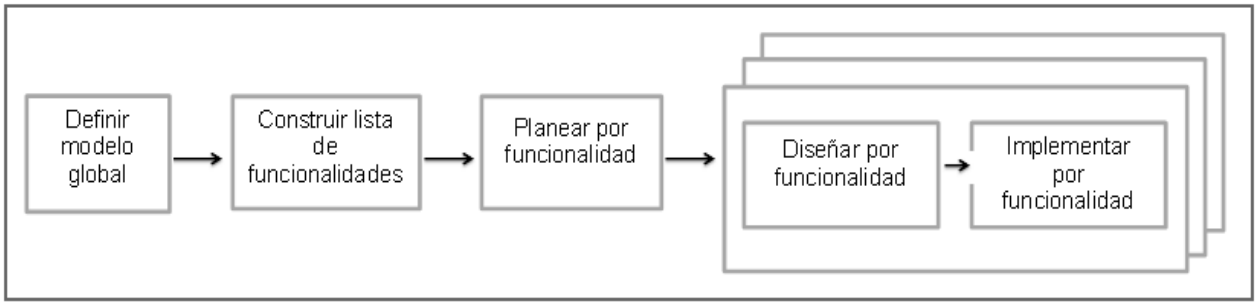


Figura 1. Fases de *FDD*. Fuente: Elaboración Propia.

Definir un modelo global

Cuando comienza el desarrollo, los expertos del dominio están al tanto de la visión, el contexto y los requisitos del sistema a construir.

Construcción de una lista de funcionalidades

Se elabora una lista de funcionalidades que resuma la funcionalidad general del sistema. La lista es elaborada por los desarrolladores y luego evaluada por el cliente. Se divide la lista en subconjuntos según la afinidad y la dependencia de las funcionalidades. La lista es finalmente revisada por los usuarios y los responsables para su validación y aprobación.

Planeación por funcionalidad

En este punto se procede a ordenar los conjuntos de funcionalidades conforme a su prioridad y dependencia, y se asigna a los programadores jefes.

Diseñar por funcionalidad

Se selecciona un conjunto de funcionalidades de la lista. Se procede a diseñar las funcionalidades, que en conjunto con el proceso de construcción por funcionalidades se realiza mediante un proceso iterativo, iteración que puede tomar de unos pocos días a un máximo de dos semanas.

Implementar por funcionalidad

Se procede a construir las funcionalidades diseñadas. Una iteración puede tomar de unos pocos días a un máximo de dos semanas. El proceso iterativo incluye inspección de diseño, codificación, pruebas unitarias e inspección de código.

Dicha metodología presenta las siguientes características (12):

- Está pensada para proyectos con tiempo de desarrollo relativamente cortos (menos de un año).
- Se basa en un proceso iterativo con ciclos cortos que producen un *software* funcional que el cliente y la dirección de la empresa pueden ver y monitorizar.
- Incluye un monitoreo constante del proyecto.
- Se obtienen resultados periódicos.

1.3.2. Lenguaje de modelado. Lenguaje Unificado de Modelado (UML)

El Lenguaje Unificado de Modelado (UML por sus siglas en inglés, *Unified Modeling Language*) es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de los sistemas de

software (13). Ofrece un estándar para describir un "plano" del sistema, incluyendo aspectos conceptuales tales como: procesos del negocio, funciones y aspectos concretos como expresiones de lenguajes de programación y esquemas de bases de datos.

1.3.3. Herramientas para el diseño. Visual Paradigm

Visual Paradigm es una herramienta *Computer Aided Software Engineering* (Ingeniería de Software Asistida por Computadoras, CASE) que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientado a objetos, construcción, pruebas y despliegue. Se escogió *Visual Paradigm* como herramienta para el diseño ya que permite dibujar todos los tipos de diagramas de clases, generar código desde diagramas y generar documentación (14).

Dicha herramienta tiene como principales características (15):

- Generación automática de la documentación del proyecto en varios formatos.
- Capacidades de ingeniería directa e inversa (código a modelo, código a diagrama).
- Disponibilidad de integrarse en los principales Entornos de Desarrollo Integrado (IDE).
- Generación de código (modelo a código, diagrama a código).
- Generación de bases de datos (transformación de diagramas de Entidad-Relación en tablas de bases de datos).
- Herramienta multiplataforma.

1.3.4. Lenguaje de programación. CSharp (C#)

Un lenguaje de programación es aquel elemento dentro de la informática que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que pone a disposición del programador para que este pueda comunicarse con los dispositivos *hardware* y *software* existentes (16).

C# se ha diseñado para compilar diversas aplicaciones que se ejecutan en .NET *Framework* (17). El mismo es utilizado como lenguaje de programación, ya que presenta varias características que a continuación se mencionan:

- Admite los conceptos de encapsulación, herencia y polimorfismo.
- Una clase puede heredar directamente de una clase primaria, pero puede implementar cualquier número de interfaces.
- Los métodos abstractos que reemplazan a los métodos virtuales en una clase primaria requieren la palabra clave *override* como medio para evitar redefiniciones accidentales.

Este lenguaje facilita el desarrollo de componentes de *software* a través de varias construcciones de lenguaje, entre las que se incluyen (18):

- Propiedades, que actúan como descriptores de acceso para variables miembro privadas.
- Atributos que proporcionan metadatos declarativos sobre tipos en tiempo de ejecución.
- Consultas Integradas al Lenguaje (*Language-Integrated Query*, LINQ) que proporcionan funciones de consulta integradas en una gran variedad de orígenes de datos.

1.3.5. Entorno integrado de desarrollo. Visual Studio .NET

Visual Studio .NET es una colección completa de herramientas y servicios, que permite crear una gran variedad de aplicaciones web, aplicaciones para escritorio y aplicaciones móviles (19).

La nueva versión de *Visual Studio* 2013 incluye nuevas funciones como *Browser Link* (Enlace del Navegador), para sincronizar los cambios entre el editor de código y las ventanas de explorador. Las herramientas web integradas de *Visual Studio*, permiten crear aplicaciones para la web y la nube basadas en estándares actuales o de próxima generación como HTML5[3] y CSS3[4], que se adaptan a exploradores tradicionales, modernos y móviles. El mismo incluye compatibilidad total con *JavaScript* para la creación de aplicaciones web interactivas (20).

1.3.6. Tecnologías a usar

.NET *Framework* 4.5

.NET *Framework* es una tecnología que admite la compilación y ejecución de aplicaciones y servicios web XML[5] (21). Consta de dos componentes principales: *Common Language Runtime* (CLR), que es el motor de ejecución que controla las aplicaciones en ejecución, y la biblioteca de clases de .NET *Framework*, que proporciona una biblioteca de código probado y reutilizable al que pueden llamar los desarrolladores desde sus propias aplicaciones.

Ofrece a las aplicaciones en ejecución algunos servicios dentro de los cuales se encuentran (22):

[3] HTML (*HyperText Markup Language*, Lenguaje de Marcas de Hipertexto): Usado para estructurar y presentar el contenido para la web.

[4] CSS (*Cascading Style Sheets*, Hojas de Estilo en Cascada): Permiten aplicar estilos y formato a una página HTML.

[5] Servicio web XML: Entidad programable que proporciona un elemento determinado de funcionalidad, como lógica de la aplicación (51).

- *Frameworks* y tecnologías de desarrollo: incluye bibliotecas para determinadas áreas de desarrollo de aplicaciones, como ASP.NET para aplicaciones web, ADO.NET para el acceso a los datos y *Windows Communication Foundation* (WCF) para las aplicaciones orientadas a servicios.
- Biblioteca de clases extensa: en lugar de tener que escribir cantidades extensas de código para controlar operaciones comunes de programación de bajo nivel, los programadores pueden usar una biblioteca de tipos accesible en todo momento y sus miembros desde la biblioteca de clases de .NET *Framework*.

JavaScript

JavaScript es un lenguaje de programación que se puede utilizar para incorporar interactividad a las páginas web. Permite crear una interfaz de usuario activo, dando información a los usuarios a medida que navega en el sitio web. También puede leer y escribir *cookies*[6], operación que hasta hace poco únicamente podía desarrollar el servidor web (23).

Bootstrap

Bootstrap simplifica el proceso de creación de diseños web combinando CSS y *JavaScript*, se puede crear interfaces que se adapten a los distintos navegadores, apoyándose en un *framework* con numerosos componentes webs.

Dentro de sus principales características se encuentra:

- Permite crear interfaces que se adapten a los diferentes navegadores, tanto de escritorio, como tablets y móviles a distintas escalas y resoluciones.
- Ofrece un diseño sólido usando estándares como CSS3 y HTML5 (24).
- Ofrece una gran variedad de componentes de interfaz de usuario, como menús desplegables, grupos de botones e iconos (25).

1.3.7. Acceso a datos

Sistema gestor de base de datos. *PostgreSQL* 9.2

Este es un sistema de gestión de base de datos (SGBD) objeto-relacional. Es mantenido por la organización *PostgreSQL Global Development Team* (Grupo Global de Desarrollo de *PostgreSQL*) y existe además una amplia comunidad de usuarios y programadores que colaboran activamente. Dicho

[6] *Cookies:* Pequeña información enviada por un sitio web, y almacenada en el navegador del usuario, de manera que el sitio web puede consultar la actividad previa del usuario.

sistema se encuentra disponible en un amplio rango de plataformas. Utiliza el modelo cliente-servidor y tiene buen desempeño con grandes volúmenes de información (26).

Características principales de su versión 9.2:

- Durabilidad: propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema (27).
- Soporte para distintos tipos de datos, como son: datos de tipo fecha, datos sobre redes (MAC, IP).
- Soporta el uso de índices, reglas y vistas.
- Soporta totalmente el acceso y procedimientos de base de datos en Java, Python, Perl y PHP (28).

Además, cuenta con una intuitiva herramienta de administración con interfaz gráfica llamada pgAdmin, la cual proporciona una forma fácil para ejecutar tareas de configuración, creación de bases de datos, tablas y usuarios.

Framework* de mapeo. *NHibernate

Es un *framework* de mapeo objeto-relacional (*Object Relational Mapping*, ORM[7]) que facilita el mapeo de los atributos entre una base de datos relacional y el modelo de los objetos de una aplicación, permitiendo además establecer estas relaciones. *NHibernate* es la conversión de *Hibernate* del lenguaje Java a C#, para su integración en la plataforma .NET. *NHibernate* se comunica con la base de datos y realiza diferentes acciones requeridas por los objetos persistentes dígase: inserción, actualización, borrado y selección (29).

1.3.8. Sistema a utilizar en la implementación de la solución. Panel de Administración de Dispositivo (Device Grid Manager, DGM)

El DGM es un sistema que permite la interacción y control centralizado de los dispositivos de *hardware*. Este sistema cuenta con tres subsistemas que interactúan como un todo: el servicio local para el manejo de dispositivos, el *framework JavaScript* DGMJS y el sistema para el control centralizado de dispositivos.

- El servicio local gestiona las peticiones que provienen desde el navegador web que han sido solicitadas por el *framework* DGMJS y reenvía estas peticiones al motor de manejo de los servicios para cuando se reciba una respuesta por parte del servicio que maneja el dispositivo, enviársela al navegador web. El servicio local es quien contiene además al grupo de clientes y controladores, los clientes constituyen la interfaz para el uso de los dispositivos, quienes utilizan los controladores que son los que interactúan directamente con el dispositivo (30).

[7] ORM: *Framework* encargado de adaptar el mundo de objetos al relacional.

Dicho sistema cuenta con 10 servicios, dentro de los cuales son de interés para el equipo de desarrollo:

- ***Encoder:*** para generar los *tokens*[8] y la activación de la pc.
- ***Notification:*** notifica al sistema la existencia de algún cambio producido en el puesto de trabajo.
- ***PcInfo:*** brinda información de cada una de las estaciones de trabajo.

[8] *Token* o Componente Léxico: Cadena de caracteres que tiene un significado coherente en cierto lenguaje de programación (52).

1.4. Conclusiones parciales

- El estudio de los referentes teóricos permitió el entendimiento de los conceptos esenciales que se relacionan con la problemática existente dentro del objeto de estudio definido.
- El análisis de las herramientas para el control de *hardware* sirvió de apoyo para definir las características y requisitos con que debe contar el sistema a desarrollar.
- La metodología seleccionada permitirá guiar el proceso de desarrollo de la aplicación.
- La definición del ambiente de desarrollo posibilitó establecer las herramientas y tecnologías para desarrollar la solución que se propone.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

Después de realizada la fundamentación teórica de la investigación, en este capítulo se presenta una descripción de la propuesta de solución, que siguiendo la metodología definida para su desarrollo permite realizar la planificación por etapas, la especificación de sus funcionalidades y el diseño del sistema mediante el modelado de los diagramas del diseño.

2.1. Propuesta de solución

En todo proceso de desarrollo es indispensable tener presente la visión y el alcance del producto a desarrollar para mantenerlo enfocado a las necesidades a resolver. Luego de realizar un análisis de la situación problemática existente, se decidió implementar un sistema con el objetivo de perfeccionar el control de *hardware* y medios básicos realizado actualmente en el CISED. A continuación (Ver **Figura 2**) se muestra el flujo de actividades que se propone para la solución propuesta.

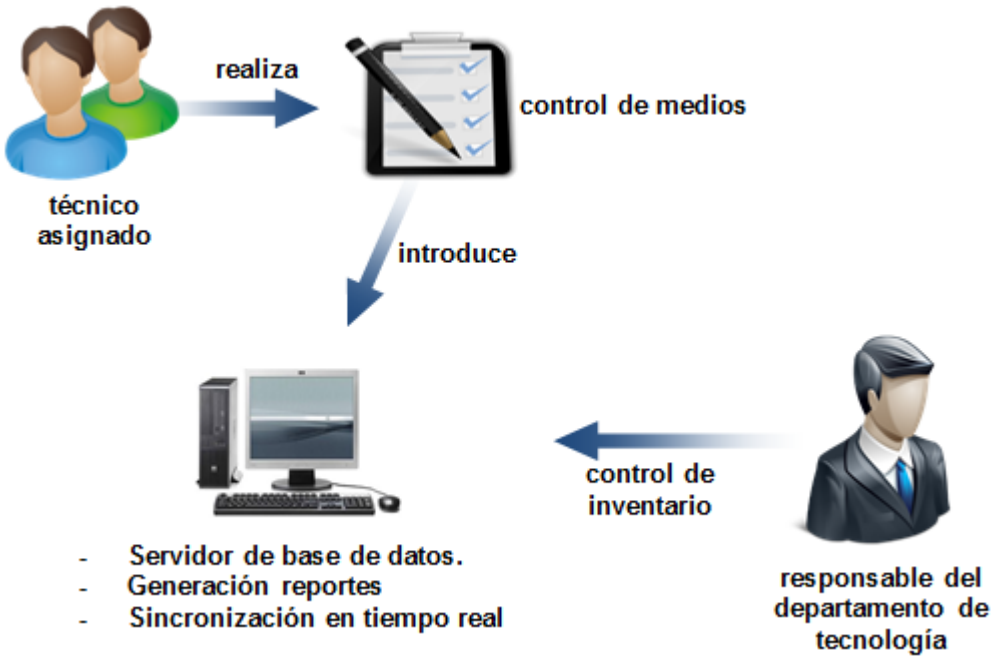


Figura 2. Propuesta de Solución. Fuente: Elaboración Propia.

Para dar cumplimiento a los objetivos planteados al inicio del trabajo se propone como solución el desarrollo de una aplicación web desarrollada sobre ASP.NET Modelo-Vista-Controlador (MVC) 4.0. Dicha aplicación estará compuesta por el Módulo Administración, el cual se encargará del control de *hardware* de cada ordenador; y la gestión y verificación de cada uno de los medios básicos existentes en cada local

del centro. Además se podrá obtener los reportes necesarios, permitiendo visualizar el estado en que se encuentra cada uno de los recursos materiales.

2.2. Modelo de dominio

Un modelo de dominio representa los conceptos más importantes dentro del dominio del problema y las relaciones entre ellos. Este modelo se realiza con el fin de proveer un marco general dentro del cual evoluciona el proyecto. Los objetos del dominio representan los eventos que suceden en el entorno en el que trabaja el sistema. Muchos de los objetos del dominio o clases pueden obtenerse mediante una especificación de requisitos o entrevistas con los expertos del dominio (31). (Ver **Figura 3**).

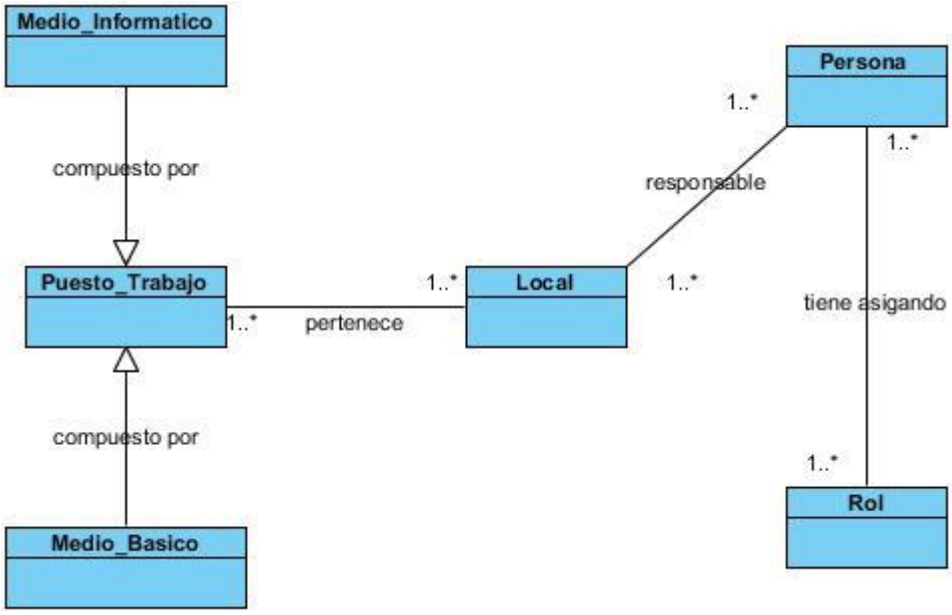


Figura 3. Modelo de Dominio. Fuente: Elaboración Propia.

Puesto de trabajo: lugar o área ocupada por una persona dentro de una organización, empresa o entidad donde se desarrollan una serie de actividades las cuales satisfacen expectativas, que tienen como objetivo, garantizar productos, servicios y bienes en un marco social (32).

Local: aquello perteneciente a un lugar o territorio; municipio o región (33).

Persona: persona que utiliza un dispositivo o un ordenador y realiza múltiples operaciones con distintos propósitos (34).

Medio Material: son los medios físicos y concretos que ayudan a conseguir algún objetivo (35).

Medio Informático: cualquier recurso de *hardware* o *software* que es usado en la informática para dar soluciones a determinados problemas.

Rol: función o papel que debe cumplir alguien o algo en un momento determinado (36).

2.3. Construcción de la lista de funcionalidades

Los requisitos funcionales son aquellos que debe cumplir el sistema, desde el punto de vista de las necesidades del usuario o cliente (37). A partir de la entrevista realizada al cliente, se pudo recopilar una serie de datos específicos que contribuyeron a la definición de los requisitos funcionales y no funcionales del sistema. Esta entrevista puede ser consultada en los Anexos (Ver **Anexo 1**). A continuación se muestra la lista de funcionalidades que se definieron:

2.3.1. Requisitos funcionales

Módulo Administración

RF 1. Autenticar usuario

RF 2. Gestionar persona

2.1. Crear persona

2.2. Eliminar persona

2.3. Modificar persona

2.4. Mostrar persona

RF 3. Gestionar local

3.1. Crear local

3.2. Eliminar local

3.3. Modificar local

3.4. Mostrar local

RF 4. Gestionar puesto de trabajo

4.1. Crear puesto de trabajo

4.2. Eliminar puesto de trabajo

4.3. Modificar puesto de trabajo

4.4. Mostrar puesto de trabajo

RF 5. Gestionar medios básicos

5.1. Crear medios básicos

5.2. Eliminar medios básicos

5.3. Modificar medios básicos

5.4. Mostrar medios básicos

RF 6. Realizar movimiento de medios

RF 7. Realizar búsqueda

RF 8. Reportar incidencia

RF 9. Visualizar información del reporte

RF 10. Controlar *hardware*

2.4. Planificación de las funcionalidades

Con el fin de realizar una planificación de la construcción de la solución propuesta (Ver **Tabla 1**), se clasifican las funcionalidades identificadas anteriormente en críticas y secundarias. Las funcionalidades críticas son las que tienen mayor importancia para el cliente, ya que cubren las principales tareas o funciones con que debe contar el sistema. Las secundarias aunque tienen un impacto menor sobre la aplicación, no deben dejar de implementarse en esta versión.

Tabla 1. Clasificación de las Funcionalidades.

Módulo	Funcionalidad	Clasificación
	Autenticar usuario	Secundaria
	Gestionar persona	Crítica
	Gestionar local	Crítica
	Gestionar puesto de trabajo	Crítica
	Gestionar medios básicos	Crítica

Administración	Realizar movimiento de medios	Crítica
	Realizar búsqueda	Secundaria
	Reportar incidencia	Crítica
	Visualizar información del reporte	Crítica
	Controlar *hardware*	Crítica

2.5. Cronograma de diseño y construcción

En la siguiente tabla (Ver **Tabla 2**) se muestran las fechas de ejecución de las funcionalidades en las fases de diseño y construcción, además de la iteración a la que pertenecen.

Tabla 2. Cronograma de Diseño y Construcción.

Módulo	**Iteración**	**Funcionalidad**	**Fecha de ejecución**	
			Diseño de la funcionalidad	**Construcción de la funcionalidad**
Administración	**1**	Autenticar usuario	25-01-2014/28-01-2014	02-02-2014/07-02-2014
		Gestionar local	25-01-2014/1-02-2014	08-02-2014/15-02-2014
		Gestionar medios básicos	25-01-2014/1-02-2014	16-02-2014/23-02-2014
		Gestionar persona	25-01-2014/1-02-2014	24-02-2014/03-03-2014
		Gestionar puesto de trabajo	25-01-2014/1-02-2014	04-03-2014/11-03-2014
		Realizar movimiento de medios	25-01-2014/1-02-2014	12-03-2014/19-03-2014

		Realizar búsqueda	25-01-2014/1-02-2014	19-03-2014/26-03-2014
	2	Reportar incidencia	27-03-2014/31-03-2014	01-04-2014/08-04-2014
		Visualizar información del reporte	27-03-2014/31-03-2014	09-04-2014/16-04-2014
		Controlar *hardware*	27-03-2014/31-03-2014	17-04-2014/25-04-2014

2.6. Descripción de las funcionalidades

En las tablas que a continuación se muestran (Ver **Tabla 3, 4 y 5**), se especifican las descripciones de algunas de las funcionalidades definidas, las demás especificaciones pueden ser consultadas en los anexos (Ver **Anexo 2**).

Tabla 3. Descripción de la Funcionalidad Autenticar Usuario.

Nombre de la Funcionalidad: Autenticar usuario	**Identificador:** RF 1
Objetivo de la Funcionalidad: Verificar que el usuario tenga acceso al sistema	
Usuario: Administrador o Técnico	
Iteración: 1	**Clasificación:** Secundaria
Precondiciones:	
Descripción: La persona introduce su usuario y contraseña en el sistema, este verifica que los datos introducidos sean correctos. • Si son correctos le permite el acceso. • Si no son correctos muestra un mensaje de información “El usuario y la contraseña introducidos son incorrectos”.	
Validaciones: Validar que no existan campos vacíos.	
Prototipo de interfaz de usuario:	

Tabla 4. Descripción de la Funcionalidad Realizar Movimiento de Medios.

Nombre de la Funcionalidad: Realizar movimiento de medios	**Identificador:** RF 6
Objetivo de la Funcionalidad: Realizar el movimiento de medios de un local para otro	
Usuario: Administrador	
Iteración: 1	**Clasificación:** Crítica
Precondiciones:	
Descripción: Se accede al Módulo de Administración y el sistema muestra entre sus funcionalidades Realizar Movimiento de Medios, el usuario selecciona esta opción y el sistema muestra la interfaz que permite introducir los datos pertinentes para realizar el movimiento de medios: • Identificador del movimiento • Estado del movimiento • Fecha del movimiento • Persona • Medio Informático • Medio Material • Nombre del local Si se selecciona Aceptar se realiza el cambio, siendo el mismo guardado en la base de datos.	
Validaciones:	
Prototipo de interfaz de usuario:	

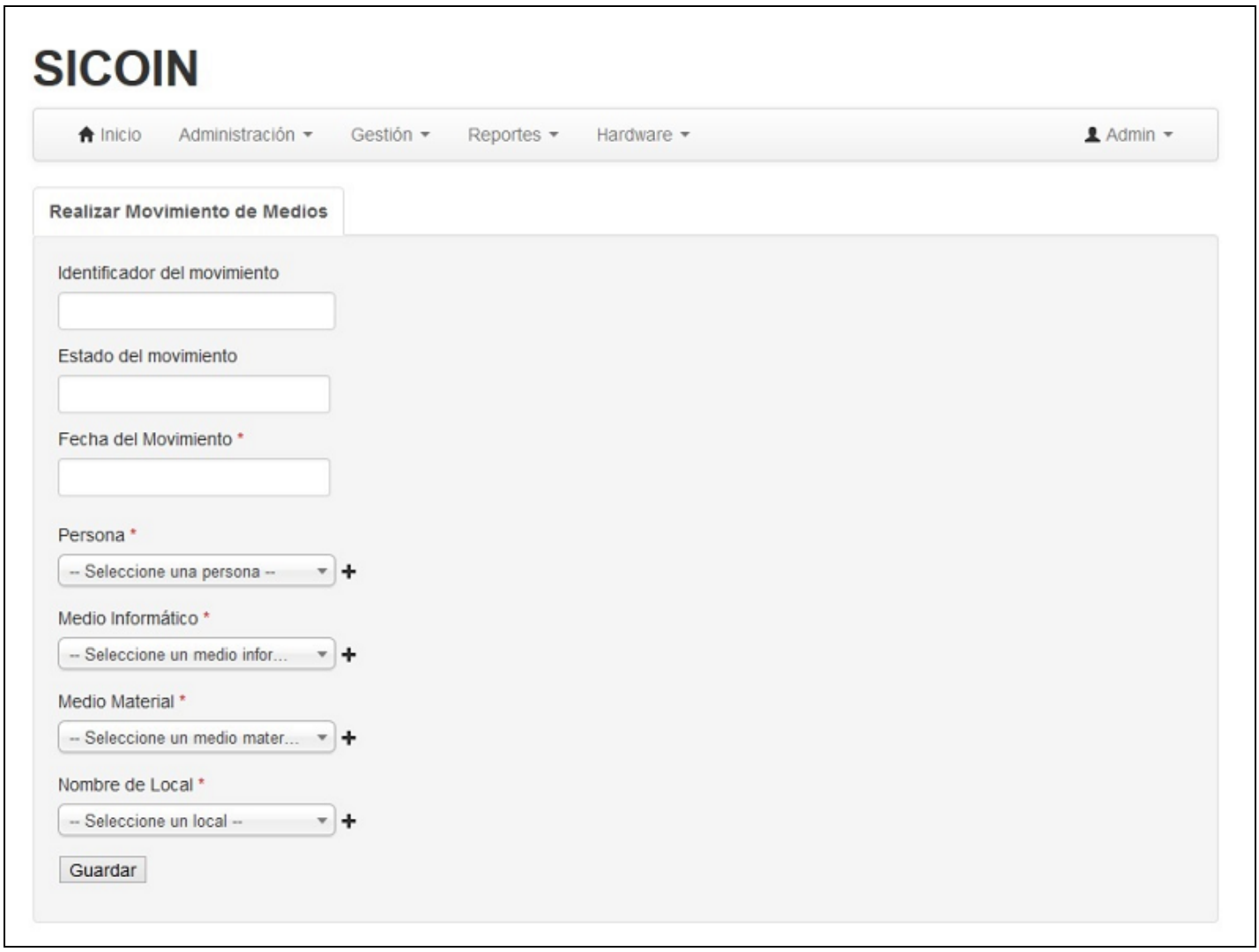


Tabla 5. Descripción de la Funcionalidad Reportar Incidencia.

Nombre de la Funcionalidad: Reportar incidencia	**Identificador:** RF 8
Objetivo de la Funcionalidad: Reportar diariamente las incidencias existentes en cada local	
Usuario: Técnico	
Iteración: 2	**Clasificación:** Crítica
Precondiciones:	
Descripción: El técnico accede al sistema, selecciona la opción de Reportar Incidencia y el sistema muestra la planilla a llenar con los datos correspondientes (Ver **Anexo 3**). Se introducen los datos y se envía la incidencia.	

Validaciones:
Prototipo de interfaz de usuario:

2.7. Requisitos no funcionales

Los requisitos no funcionales son las propiedades que el sistema debe cumplir. A continuación se definen estos para el sistema que se desea implementar:

- **Requisitos de *Software***

RNF 1. El sistema operativo que debe estar instalado en la estación de trabajo es Windows XP o superior.
RNF 2. *Framework* .NET 4.5.

- **Requisitos de *Hardware***

RNF 3. Las estaciones de trabajo del cliente deben tener como mínimo 512 Mb de RAM.
RNF 4. El servidor para la instalación del sistema para el control centralizado de dispositivos debe tener como mínimo 4 Gb de RAM, 20 Gb de disco duro disponible para el almacenamiento de la base de datos y procesador superior a 2.40 GHz.

- **Requisitos de implementación**

RNF 6. El sistema debe implementarse usando el lenguaje C# sobre la plataforma ASP.NET.

RNF 7. El sistema gestor de bases de datos será *PostgreSQL* 9.2.

RNF 8. El sistema debe desarrollarse usando el IDE *Visual Studio 2013.*

- **Requisitos de Seguridad**

RNF 9. Autenticación segura para acceder a la aplicación.

RNF 10. La seguridad se define a nivel de roles, con el fin de mantener la integridad de los datos en función del acceso de cada uno de ellos y de proteger la información.

- **Requisitos de Usabilidad**

RNF 11. El sistema debe presentar una interfaz que permita al usuario una fácil interacción.

RNF 12. El sistema podrá ser utilizado por cualquier usuario que presente las siguientes características:

- Conocimientos básicos relativos al uso de una computadora.

- Conocimientos básicos del sistema operativo Windows.

2.8. Especificación de la arquitectura de *software*

La arquitectura de *software*, se define como la organización fundamental de un sistema representada en sus componentes, las relaciones entre ellos, el ambiente y los principios que orientan su diseño y evolución. Los Patrones Arquitectónicos son los que definen la estructura de un sistema, tienen una serie de directivas para organizar los componentes, con el objetivo de facilitar la tarea del diseño de dicho sistema (38).

Se definió que el sistema se desarrolle a través de la arquitectura cliente-servidor; la cual posibilita la relación entre dos aplicaciones, donde una realiza una petición y la otra devuelve una respuesta.

2.8.1. Patrón de arquitectura MVC 4.0

El Modelo Vista Controlador (MVC) es un patrón de arquitectura de *software* que separa los datos y la lógica de negocio de una aplicación, de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones (39). (Ver **Figura 4**).

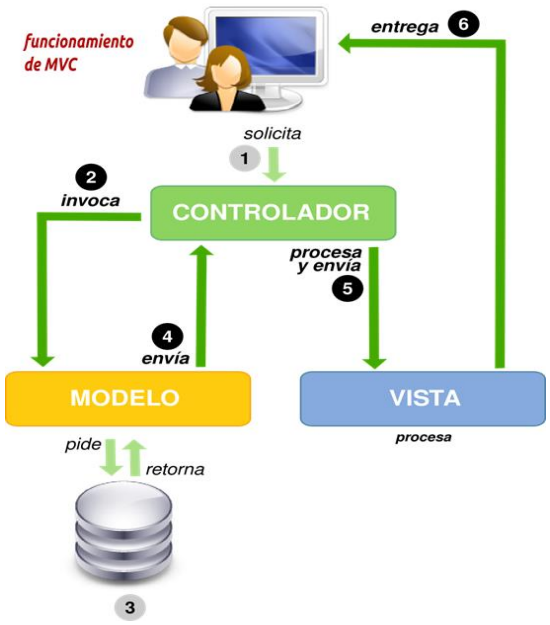


Figura 4. Patrón Arquitectónico Modelo-Vista-Controlador. Fuente: Elaboración Propia.

MVC separa la interfaz de usuario de una aplicación en tres aspectos principales (40):

- **El Modelo:** trabaja con los datos, por tanto contiene mecanismos para acceder a la información y también para actualizar su estado. Los datos los tendremos en una base de datos, por lo que en

los modelos tendremos todas las funciones que accederán a las tablas y harán los correspondientes insertar, seleccionar, actualizar y eliminar.

- **La Vista:** envía las acciones del usuario al controlador, además de permitir al controlador seleccionar las vistas pertinentes. Contienen el código de la aplicación que va a producir la visualización de las interfaces de usuario.
- **El Controlador:** define el comportamiento de la aplicación, contiene el código necesario para responder a las acciones que se solicitan, como visualizar un elemento y una búsqueda de información. Sirve de enlace entre las vistas y los modelos.

Una aplicación basada en ASP.NET MVC presenta varias ventajas entre las que se encuentran:

- Facilita la administración de la complejidad, al dividir una aplicación en el modelo, la vista y el controlador.
- Plantillas de proyecto predeterminadas, renovadas y modernizadas.
- Soporte de aplicaciones móviles (41).
- Al tener la lógica separada de la interfaz es mucho más sencillo crear las pruebas unitarias y desarrollar mediante *Test Driven Development* (Desarrollo Basado en Pruebas, *TDD*).
- Simplicidad con la que se puede gestionar y mantener el sistema, así como la posibilidad de trabajar en paralelo, es decir puedes tener diseñadores trabajando en las vistas mientras que los desarrolladores se pueden centrar en el controlador y el modelo (42).

2.9. Distribución lógica del sistema

El sistema se encuentra conformado por las capas que a continuación se detallan, orientadas al desarrollo de soluciones sobre la arquitectura cliente-servidor. (Ver **Figura 5**).

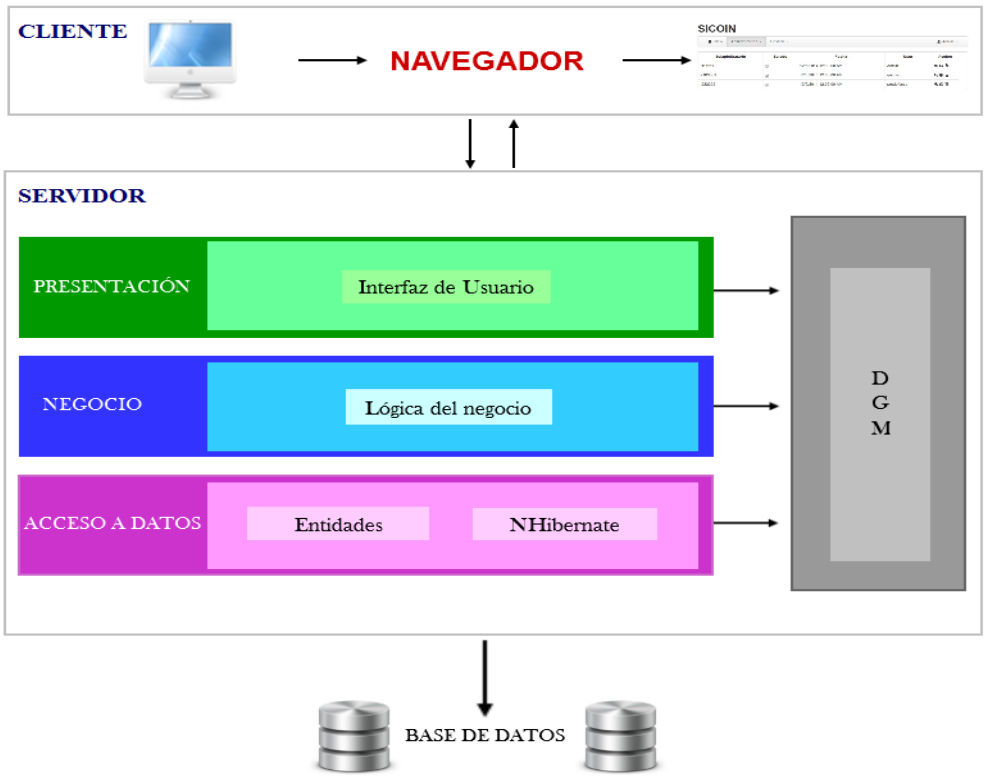


Figura 5. Vista Lógica de la Arquitectura. Fuente: Elaboración Propia.

- **Capa de Presentación**

Está compuesta por todas las interfaces de usuario y los componentes necesarios para su correcto funcionamiento. Es la capa donde el sistema interactúa con el usuario.

- **Capa de Negocio**

En esta capa se recogen todas las funcionalidades necesarias para darle solución a los requisitos planteados.

- **Capa de Acceso a Datos**

Trabaja directamente con la fuente de datos establecida. En esta capa se encuentra incluido el ORM *NHibernate* utilizado para la generación del modelo de datos y se encarga de la manipulación de la información en la base de datos.

- **DGM**

Capa transversal la cual va a permitir la interacción y control centralizado de los dispositivos de *hardware*.

2.10. Diagramas de clases del diseño

Los diagramas de clases son los más utilizados en el modelado de sistemas orientados a objetos, puesto que muestran un conjunto de clases e interfaces, así como sus relaciones (43). Se utilizan para modelar la vista estática del diseño del *software*, especificar y documentar modelos estructurales, aplicando ingeniería directa e inversa. En los Anexos (Ver **Anexo 4**) se muestra el diagrama de clases del diseño del sistema.

2.11. Modelo de datos

Un modelo de datos describe la representación lógica y física de los datos persistentes usados por el sistema (44). En los Anexos (Ver **Anexo 5**) se muestra el modelo de datos del sistema.

2.11.1. Descripción del modelo de datos

El modelo de datos utilizado en la aplicación contiene 21 tablas. A continuación (Ver **Tabla 6**) se describirán las mismas.

Tabla 6. Composición de Tablas del Modelo de Datos.

Nombre de la tabla	Descripción
Persona	Contiene los datos de cada persona existente en el sistema.
Usuario	Usuario que tiene permiso para acceder al sistema.
Rol	Contiene los roles que se le asignarán a cada usuario del sistema.
Permiso_Rol	Nomenclador que contiene los permisos correspondientes a cada rol.
Local	Contiene los locales existentes en el centro.
Puesto_Trabajo	Almacena los puestos de trabajo existentes en cada local.
Medio_Básico	Contiene los medios básicos existentes en el centro, así como el puesto de trabajo al que pertenece.
Medio_Informático	Registra los datos de cada medio informático existente en el centro, así como el puesto de trabajo al que pertenece.
Movimiento	Contiene los datos de cada movimiento realizado en el sistema.

Display[9]	Contiene los datos de cada *display* existente, así como el puesto de trabajo al que pertenece.
Batería	Contiene los datos de cada batería existente, así como el puesto de trabajo al que pertenece.
Teclado	Contiene los datos de cada teclado existente, así como el puesto de trabajo al que pertenece.
Mouse[10]	Contiene los datos de cada *mouse* existente, así como el puesto de trabajo al que pertenece.
CPU	Contiene los datos de cada *cpu* existente, así como el puesto de trabajo al que pertenece.
HDD[11]	Contiene los datos de cada *hdd* existente, así como la CPU a la que pertenece.
Memoria	Contiene los datos de cada memoria existente, así como la CPU al que pertenece.
Fuente	Contiene los datos de cada fuente existente, así como la CPU a la que pertenece.
Motherboard[12]	Contiene los datos de cada *motherboard* existente, así como la CPU a la que pertenece.
Unidad_Optica	Contiene los datos de cada unidad óptica existente, así como la CPU a la que pertenece.
Tarjeta	Contiene los datos de cada tarjeta existente, así como la CPU a la que pertenece.
Microprocesador	Contiene los datos de cada microprocesador existente, así como la CPU a la que pertenece.

[9] *Display* o Pantalla: Aparato electrónico que muestra visualmente la información contenida en él.

[10] *Mouse* o Ratón: Dispositivo utilizado para facilitar el manejo de un entorno gráfico en una computadora.

[11] *Hard Disk Drive* o Disco Duro: Dispositivo de almacenamiento de datos.

[12] *Motherboard* o Plaqueta Madre: Tarjeta de circuito impreso que permite la integración de todos los componentes de una computadora (microprocesador, la memoria).

2.12. Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular. El mismo identifica: clases, instancias, roles, colaboraciones y la distribución de responsabilidades (45).

Dentro de los patrones de diseño se encuentran dos grupos fundamentales conocidos como Patrones Generales de Asignación de Responsabilidades de *Software* (*General Responsibility Assignment Software Patterns,* GRASP) y Banda de los Cuatro (*Gang Of Four*, GOF).

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos. Mientras que los patrones GOF describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros (46).

A continuación se analizan los patrones utilizados para la implementación de la solución.

2.12.1. Patrón GRASP

- **Patrón Experto**

¿Quién asumirá la responsabilidad en el caso general?

Consiste en asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. Con este patrón se pretende que los objetos realicen acciones relacionadas con la información que poseen (47).

En la clase Persona que a continuación se representa se hace referencia a una lista de usuarios y movimientos, que no son más que objetos que pertenecen a otras clases desarrolladas y que mediante esta clase Persona pueden ser relacionados para satisfacer un conjunto de necesidades de acuerdo a la información requerida.

```
public class Persona {
    0 references
    public Persona() {
        Movimientos = new List<Movimiento>();
        Usuarios = new List<Usuario>();
    }
    [Required(ErrorMessage = "El campo id es requerido")]
    [RegularExpression("^[0-9]*$", ErrorMessage = "solo números")]
    14 references
    public virtual string IdPersona { get; set; }
    [Required(ErrorMessage = "El campo primer apellido es requerido")]
    [RegularExpression("^[a-z A-z]*$",ErrorMessage = "Solo letras")]
    5 references
    public virtual string PrimerApellido { get; set; }
    5 references
    public virtual Local Local { get; set; }
    [Required(ErrorMessage = "El campo segundo apellido es requerido")]
    [RegularExpression("^[a-z A-z]*$", ErrorMessage = "Solo letras")]
    5 references
    public virtual string SegundoApellido { get; set; }
    [Required(ErrorMessage = "El campo carnet de identidad es requerido")]
    [RegularExpression("^[0-9]*$", ErrorMessage = "solo números")]
    [StringLength(11, MinimumLength = 11, ErrorMessage = "Debe tener 11 caracteres")]
    4 references
    public virtual string CarneIdentidad { get; set; }
    [Required(ErrorMessage = "El campo nombre es requerido")]
    [RegularExpression("^[a-z A-z]*$", ErrorMessage = "Solo letras")]
    6 references
    public virtual string Nombre { get; set; }
    2 references
    public virtual IList<Movimiento> Movimientos { get; set; }
    2 references
    public virtual IList<Usuario> Usuarios { get; set; }
```

Figura 6. Ejemplo de Patrón Experto.

- **Patrón Controlador**

¿Quién administra un evento del sistema?

Se encarga de asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. Un evento del sistema es una acción generada por un actor externo; es un evento de entrada externa (46).

En la clase *LocalController* que hereda de la clase *BaseController* se evidencia cómo se otorgan permisos a roles específicos para que puedan acceder a determinadas funcionalidades implementadas con anterioridad. Como es el ejemplo de la gestión de locales.

```
public class LocalController : BaseController
{
    [Authorize(Roles = PermisoRols.AgregarLocal)]
    0 references
    public ActionResult Agregar(string id)...
    [HttpPost]
    0 references
    public ActionResult Agregar(Local local)...

    [Authorize(Roles = PermisoRols.ListarLocal)]
    0 references
    public ActionResult ListarLocal()...

    [Authorize(Roles = PermisoRols.VerDetallesLocal)]
    0 references
    public ActionResult VerDetallesLocal(string id)...
    0 references
    public static Local Local(string id)...

    [Authorize(Roles = PermisoRols.EditarLocal)]
    0 references
    public ActionResult EditarLocal(string id)...
    [HttpPost]
    0 references
    public ActionResult EditarLocal(Local local)...

    [Authorize(Roles = PermisoRols.EliminarLocal)]
    0 references
    public ActionResult EliminarLocal(string id)...
```

Figura 7. Ejemplo de Patrón Controlador.

2.12.2. Patrón GOF

- **Patrón Singleton**

Está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Consiste en garantizar que una clase solo tenga una instancia y proporcionar un punto de acceso global a ella (48).

```
public class NHibernateSessionManager
{
    private static ISessionFactory _sessionFactory;
    public static readonly NHibernateSessionManager Instance = new NHibernateSessionManager();

    100 references
    public ISession GetSession
    {
        get { return _sessionFactory.OpenSession(); }
    }
```

Figura 8. Ejemplo de Patrón Singleton.

2.13. Conclusiones parciales

- La metodología seleccionada posibilitó guiar el proceso de desarrollo del sistema.
- La definición de las funcionalidades del sistema permitieron tener una visión general del mismo y realizar una planificación detallada de cada una de las etapas en correspondencia con el tiempo estimado. Mientras que su especificación permitió conocer al detalle la descripción de cada una de ellas.
- La elaboración del modelo de datos hizo posible tener una visión general de la estructura de la base de datos del sistema y el uso de los patrones de diseño permitió definir la estructura general del sistema a desarrollar.
- Con este capítulo se crearon las bases necesarias para la posterior implementación del sistema.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

En este capítulo se presentan los estándares de codificación utilizados, se muestra la distribución física del sistema entre los diferentes nodos a través del diagrama de despliegue y por último, se presentan los resultados de las principales pruebas funcionales realizadas al sistema para verificar su correcto funcionamiento.

3.1. Estándares de codificación

Un estándar de codificación comprende todos los aspectos de la generación de código. El uso de estándares de codificación mejora la interpretación del código de programación entre los integrantes del equipo de desarrollo (49). A continuación se reflejan los estándares definidos por el equipo de desarrollo para la propuesta solución:

Convenciones de diseño:

- Se escribe únicamente una instrucción por línea.
- Se agrega al menos una línea en blanco entre las definiciones de método y las definiciones de propiedad.
- Se usan paréntesis para dotar a las cláusulas de un formato de expresión.

Convenciones de los comentarios:

- El comentario se sitúa en una línea independiente, no al final de una línea de código.
- No se crean bloques de asteriscos (*) con formato alrededor de los comentarios.

Convenciones del lenguaje:

- Se utilizan instrucciones try-catch para el control de excepciones.

Ejemplos:

```
public ActionResult Details(string id)
{
    if (id == null)
    {
        return HttpNotFound();
    }
    else
    {
        var session = NHibernateSessionManager.Instance.GetSession;
        return View(session.Query<Usuario>().Single(x=>x.SolapinUsuario == id));
    }

}
```

Figura 9. Estándar de Codificación. Convenciones de diseño.

```
public ActionResult Create(Usuario usuario)
{
    try
    {
        // TODO: Add insert logic here
        var session = NHibernateSessionManager.Instance.GetSession;
        using (var transaction=session.BeginTransaction())
        {
            try
            {
                session.Save(usuario);
                transaction.Commit();
            }
            catch (Exception exception)
            {
                transaction.Rollback();
                throw new Exception(exception.Message,exception.InnerException);
            }
        }

        return RedirectToAction("List");
    }
    catch
    {
        return View();
    }
}
```

Figura 10. Estándar de Codificación. Convenciones de los comentarios y lenguaje.

3.2. Tratamiento de errores

Para el tratamiento de errores se valida cada uno de los datos introducidos al sistema por el usuario. Además, se verifican los campos obligatorios y se revisa el tipo de dato de cada uno. Todos los mensajes de error se muestran en color rojo para que resalten. (Ver **Figura 11**).

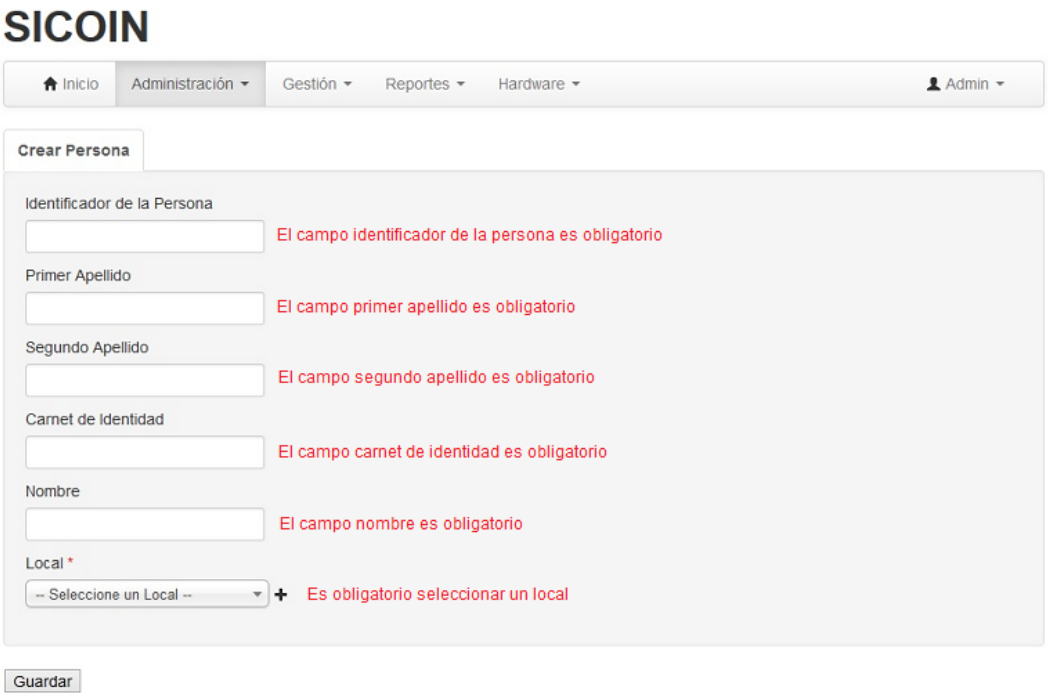


Figura 11. Ejemplo de Tratamiento de Errores.

Para la realización de las validaciones se utilizó *DataAnnotations*, la misma es una biblioteca de. NET *Framework*, que reside en *System.ComponentModel.DataAnnotations*. Contiene atributos de validación para hacer cumplir las reglas de validación, atributos de visualización para especificar cómo se muestran los datos de la clase o miembro, el modelado de datos de atributos para especificar el uso previsto de los miembros de datos y las relaciones entre las clases de datos.

3.3. Diagrama de despliegue

Un diagrama de despliegue es un modelo de objetos que describe la distribución física del sistema, en términos de cómo se distribuyen las funcionalidades entre los nodos de cómputo. Cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo similar (50).

En el diagrama que se muestra (Ver **Figura 12**), el nodo PC_Cliente representa la computadora en la cual estará desplegada la aplicación (aplicación de interfaz gráfica), además la misma contendrá el servicio local del DGM el cual posibilitará obtener la información de cada uno de los componentes que constituye

la PC. Estas aplicaciones consumen los demás servicios del DGM que se exponen en el servidor (nodo Servidor_Web) utilizando el protocolo HTTPS. En el nodo Servidor_BD se encuentran los componentes que brindan la información necesaria para implementar las funcionalidades.

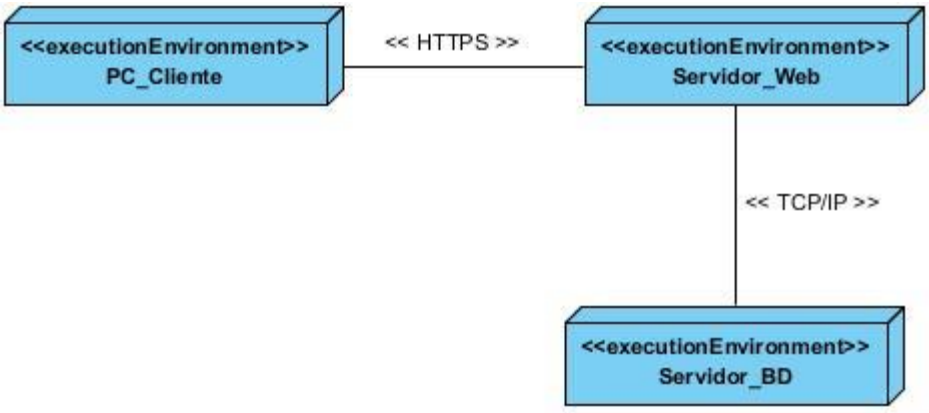


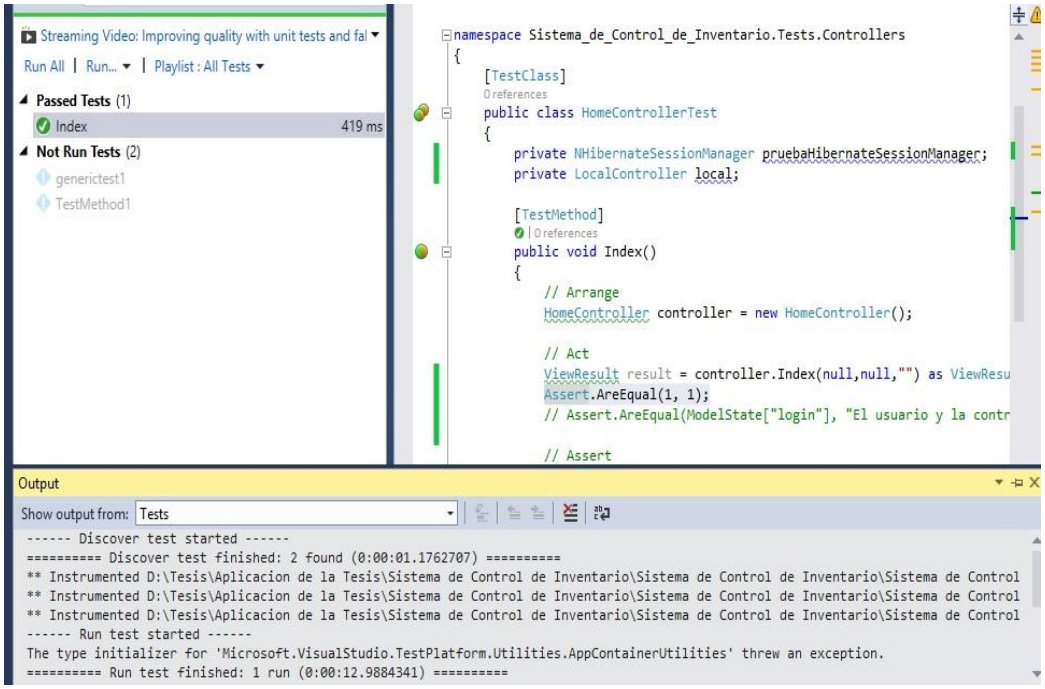
Figura 12. Diagrama de Despliegue. Fuente: Elaboración Propia.

3.4. Pruebas al sistema

Con el fin de detectar errores y verificar la calidad del *software* se realizaron las pruebas de caja blanca, las cuales permitieron verificar el funcionamiento del módulo; y las pruebas de caja negra permitieron estudiar el *software* desde el punto de vista de las entradas que recibe y las respuestas que produce, sin tener en cuenta su funcionamiento interno.

3.4.1. Pruebas unitarias. Aplicación de pruebas de caja blanca

Las pruebas unitarias son las que se realizan al código con el objetivo de detectar errores en el mismo. Estas permiten verificar a través de la ejecución de un fragmento de código si la aplicación cumple con los requisitos definidos. A continuación (Ver **Figura 13**) se muestran algunos ejemplos de pruebas unitarias realizadas:



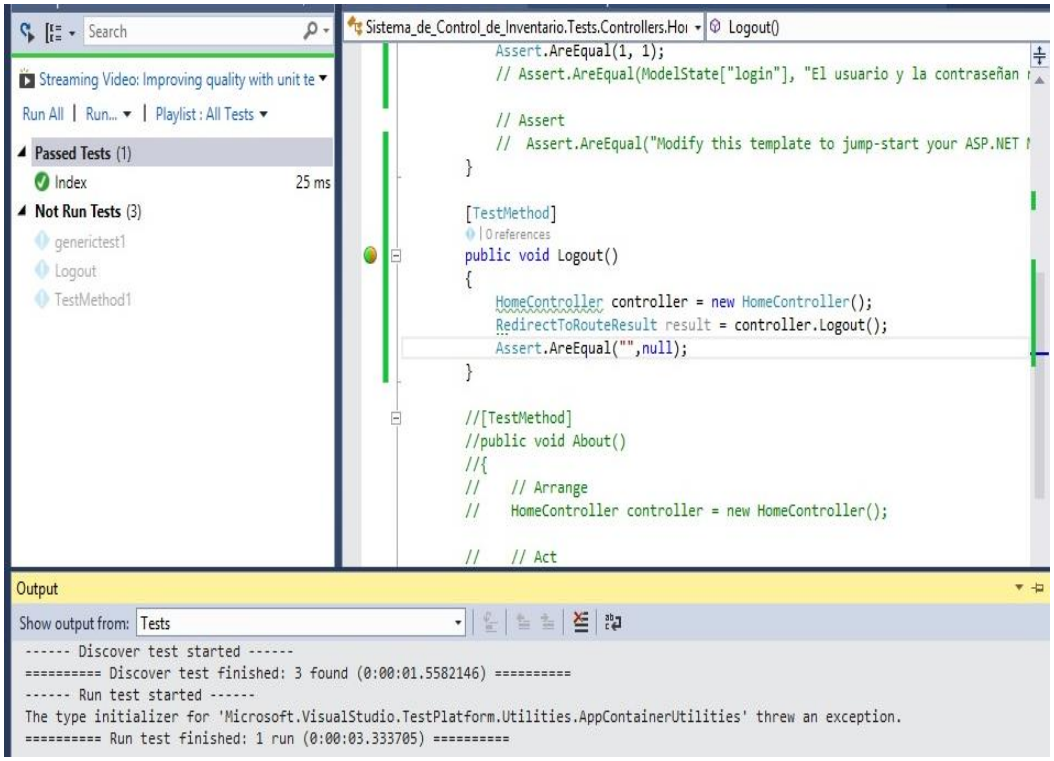


Figura 13. Ejemplos de Pruebas Unitarias.

3.4.2. Aplicación de pruebas de caja negra

Las pruebas de caja negra permiten obtener un conjunto de condiciones de entrada que ejerciten todos los requisitos funcionales del sistema. En las tablas (Ver **Tabla 7**, **8** y **9**) se muestra el diseño de algunos

de los casos de pruebas elaborados para comprobar cada una de las funcionalidades del sistema, al probar la funcionalidad con las diversas combinaciones de valores posibles de las variables. Los restantes casos de pruebas se pueden consultar en el **Anexo 6**.

Tabla 7. Caso de prueba "Autenticar Usuario".

RF	Descripción	Texto	Respuesta del sistema
RF 1. Autenticar usuario	Es necesario autenticarse para poder acceder al sistema	Usuario y Contraseña (V[13])	El sistema le permite al usuario acceder a las opciones del sistema
		Usuario y Contraseña (I[14])	El sistema muestra un mensaje de error informando que existen datos incorrectos

Tabla 8. Caso de prueba "Realizar Movimiento de Medios".

RF	Descripción	Texto	Respuesta del sistema
RF 6. Realizar movimiento de medios	Inicia cuando el administrador selecciona la opción de realizar movimiento de medios	Realizar Movimiento de Medios	El sistema realiza el movimiento de medios en dependencia de la opción seleccionada

[13] **V:** Significa que se introdujo un valor válido.

[14] I: Significa que se introdujo un valor inválido.

Tabla 9. Caso de prueba “Reportar Incidencia”.

RF	Descripción	Texto	Respuesta del sistema
RF 8. Reportar incidencia	Inicia cuando el técnico selecciona la opción de reportar incidencia	Reportar Incidencia (V)	El sistema realiza el reporte de incidencia en dependencia de la opción seleccionada
		Reportar Incidencia (I)	El sistema muestra un mensaje de error informando que se entraron datos incorrectos o campos vacíos que imposibilitaron la realización del reporte de incidencia

3.5. Resultados de las pruebas de funcionalidad

A continuación (Ver **Figura 14**) se encuentran representados los resultados de las pruebas de funcionalidad que se han realizado en cada una de las iteraciones. Luego de culminar las dos iteraciones de pruebas al sistema, se evidencia en sentido general que las Funcionalidades Incorrectas (FI) fueron totalmente corregidas, por lo que la calidad del sistema es la esperada.

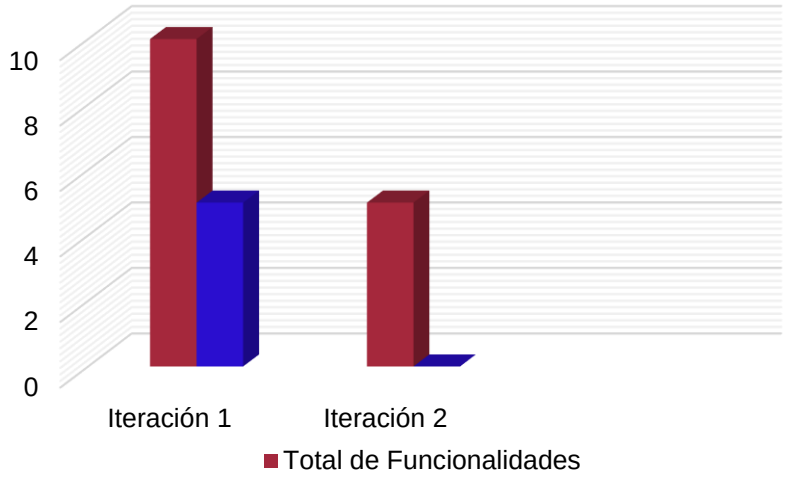


Figura 14. Gráfico del Resultados de las Pruebas de Funcionalidad. Fuente: Elaboración Propia.

3.6. Beneficios del sistema

El sistema desarrollado aporta una serie de beneficios al CISED dentro de los cuales se encuentra que:

- La información de todos los recursos del centro es administrada de manera centralizada y segura, lo que hace posible tener un mejor control de *hardware* y medios básicos del CISED.
- Reduce en un gran porciento el trabajo manual.
- Permite realizar movimientos de medios de manera informatizada.
- Genera una serie de reportes que permiten consultar toda la información existente.

3.7. Conclusiones parciales

- El uso de estándares de codificación permitió mantener uniformidad en la codificación durante la implementación del sistema.
- La realización de las pruebas de caja blanca y caja negra al sistema, permitieron encontrar una serie de funcionalidades incorrectas, las cuales fueron corregidas contribuyendo así a la calidad del producto desarrollado.

CONCLUSIONES GENERALES

- La aplicación de los métodos científicos específicamente la entrevista, permitió la identificación de los requisitos funcionales y no funcionales que dan solución a la problemática planteada y al trabajo de investigación.
- La aplicación de la metodología seleccionada hizo posible guiar todo el proceso de desarrollo de *software* y sus artefactos, necesarios para obtener un producto que satisfaga las necesidades del cliente.
- El desarrollo del sistema permite un mejor control de *hardware* y medios básicos en el CISED, así como la generación de reportes, una mayor seguridad de la información y la actualización en tiempo real de todos los recursos del centro.
- La realización de las pruebas unitarias y las pruebas de caja negra permitieron verificar el correcto funcionamiento del sistema.

RECOMENDACIONES

- Se recomienda que el uso del Sistema de Control de *Hardware* y Medios Básicos sea extendido a otras áreas de la Universidad de las Ciencias Informáticas con el fin de que obtengan un control más eficiente de sus recursos.
- Debe incluirse el análisis de elementos de *software* que contribuyan a un mayor control de todos los recursos actuales de la institución.

Referencias Bibliográficas

1. **[En línea] http://www.monografias.com/trabajos90/diseno-sistema-control-inventario/diseno-sistema-control-inventario.shtml.**

2. Definición ABC. [En línea] http://www.definicionabc.com/economia/inventario.php.

3. [En línea] http://es.scribd.com/doc/53051515/Definicion-de-inventario-fisico.

4. eumed.net. [En línea] http://www.eumed.net/cursecon/ecolat/cu/2012/cbg.html.

5. OCS Inventory Team. [En línea] http://www.ocsinventory-ng.org/en/.

6. DNA. [En línea] http://www.netsupportdna.com/es/.

7. Rodas XXI. [En línea] [Citado el: 24 de 02 de 2014.] http://www.rodasxxi.cu/rodasxxi.php.

8. [En línea] http://www.latinuxpress.com/books/drafts/cacic/caps/09.html.

9. Ing. Javier Ramirez Hernández, Ing.Lilian Alvarez Almanza, Ing.Alien Fernández Fuentes, Ing.Yuriel Martínez Ordaz. *SOLUCIÓN PARA EL CONTROL DE LOS ACTIVOS FIJOS TANGIBLES EN EL SISTEMA CEDRUX.*

10. [En línea] ftp://10.0.0.22/documentacion/Ingenieria%20Software/Metodologias/.

11. Ingeniería de Soporte Lógico. [En línea] https://sites.google.com/site/ingsoportelogico/home/feature-driven-development-fdd.

12. Pressman, Roger S. *SOFTWARE ENGINEERING: A PRACTITIONER,S APPROACH, SEVENTH EDITION.* ISBN 978-0-07-337597-7.

13. Ivar Jacobson, Grady Booch, James Rumbaugh. *El Proceso Unificado de Desarrollo de Software.* ISBN 84-7829-036-2.

14. Visual Paradigm. [En línea] [Citado el: 09 de 03 de 2013.] http://www.visual-paradigm.com/product/vpuml/.

15. Scribd. [En línea] http://es.scribd.com/doc/166415572/Visual-Paradigm.

16. definición.org. [En línea] http://www.definicion.org/lenguaje-de-programacion.

17. Microsoft. [En línea] http://msdn.microsoft.com/es-es/library/kx37x362.aspx.

18. Microsoft. [En línea] http://msdn.microsoft.com/es-es/library/z1zx9t92.aspx.

19. Microsoft. [En línea] http://msdn.microsoft.com/es-es/library/aa291755%28v=vs.71%29.aspx.

20. Visual Studio. [En línea] http://www.visualstudio.com/es-xl/products/visual-studio-professional-with-msdn-vs.aspx.

21. Microsoft. [En línea] http://msdn.microsoft.com/es-es/library/zw4w595w%28v=vs.110%29.aspx.

22. Microsoft. [En línea] http://msdn.microsoft.com/es-es/library/hh425099%28v=vs.110%29.aspx.

23. Universidad Nacional Autónoma de México. [En línea] http://recursosweb.unam.mx.

24. Rodríguez, Txema. Genbeta:dev. [En línea] http://www.genbetadev.com/frameworks/bootstrap.

25. Microsoft ASP .NET. [En línea] http://www.asp.net/visual-studio/overview/2013/creating-web-projects-in-visual-studio.

26. PostgreSQL-es. [En línea] http://www.postgresql.org.es/documentacion.

27. PostgreSQL Comunidad Técnica de Desarrollo. [En línea] http://postgresql.uci.cu/?page_id=30.

28. 2nd Quadrant Professional PostgreSQL. [En línea] http://2ndquadrant.com/es/postgresql/.

29. José M. Chinchilla, Carlos A. Martínez, Florentin Lazo Hernández, José D. Palacios Pacheco, José F. Portillo Cárcamo. [En línea] http://www.slideshare.net/DamianPalacios/motor-de-persistencia-nhibernate#.

30. Sandy Campos Rodriguez, Hector Luis Rodríguez Sánchez. Sistema para la interacción y control centralizado de dispositivos en aplicaciones web v2.0. 2012.

31. Ivar Jacobson, Grady Booch, James Rumbaugh. *El Proceso Unificado de Desarrollo de Software.* ISBN: 84-7829-036-2.

32. Puesto de Trabajo-Grupo DOS. [En línea] [Citado el: 18 de 03 de 2014.] http://pdtgrupodos.blogspot.com/2012/10/concepto-de-puesto-de-trabajo.html.

33. The Free Dictionary. [En línea] http://es.thefreedictionary.com/local.

34. Definición ABC. [En línea] http://www.definicionabc.com/tecnologia/usuario.php.

35. Definición.de. [En línea] http://definicion.de/recursos-materiales/.

36. The Free Dictionary. [En línea] http://es.thefreedictionary.com/rol.

37. Pressman, Roger S. *Ingeniería del Software. Un Enfoque Práctico. 6ta Edición.*

38. Alonso. G., Casati. F., Kuno H., Machiraju, V. *Web Services Concepts, Architectures and Applications.* 2010.

39. Microsoft. [En línea] http://msdn.microsoft.com/es-es/library/gg416514%28v=vs.108%29.aspx.

40. Alvarez, Miguel Angel. DesarrolloWeb.com. [En línea] http://www.desarrolloweb.com/articulos/que-es-mvc.html.

41. Microsoft ASP.NET. [En línea] http://www.asp.net/mvc/mvc4.

42. Portero, Víctor Fernández. Compilando.es. [En línea] http://www.compilando.es/post/2011/05/28/Arquitectura-MVC-191;como-puede-mejorar-mi-aplicacion.aspx.

43. Ledesma, Javier Darío Fernández. *Sistemas organizacionales. Teoría y práctica.* Bogotá : Universidad Cooperativa de Colombia, 2005. ISBN: 958-8325-02-6.

44. Ortiz, Antonio Moreno. *Estudios de Lingüística del Español. DISEÑO E IMPLEMENTACIÓN DE UN LEXICÓN COMPUTACIONAL PARA LEXICOGRAFÍA Y TRADUCCIÓN AUTOMÁTICA.* ISSN: 1139-8736.

45. GAMMA, E., HELM, R. y JOHNSON, R. y VLISSIDES, J. *Patrones de diseño.* 2000.

46. Larman, Craig. *UML y Patrones.* 2010.

47. —. Prácticas de Software. [En línea] http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

48. Microsoft. [En línea] http://msdn.microsoft.com/es-es/library/bb972272.aspx.

49. *Microsoft Developer Network.* [En línea] [Citado el: 19 de 03 de 2014.] http://msdn.microsoft.com/es-es/library/aa291591%28v=vs.71%29.aspx.

50. Taringa. [En línea] http://www.taringa.net/posts/apuntes-y-monografias/10119154/Proceso-Unificado-de-Desarrollo-de-Software.html. .

51. Microsoft. [En línea] http://msdn.microsoft.com/es-es/library/w9fdtx28%28v=vs.90%29.aspx.

52. C++. [En línea] http://www.zator.com/Cpp/E3_2.htm.

GLOSARIO DE TÉRMINOS

A

Arquitectura: organización fundamental de un sistema representada en sus componentes, las relaciones entre ellos, el ambiente y los principios que orientan su diseño y evolución.

B

Base de Datos: conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

F

Framework: tecnología que admite la compilación y ejecución de aplicaciones y servicios web XML.

FDD (*Feature Driven Development*): desarrollo basado en funcionalidades, es un enfoque ágil para el desarrollo de sistemas.

G

GPL (*General Public License*): el uso de la GPL ordinaria para una biblioteca la hace disponible únicamente para programas libres.

H

Hardware: parte tangible de un sistema informático.

I

IDE: *software* compuesto por un conjunto de herramientas de programación. Es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

M

MVC (Modelo Vista Controlado): patrón de arquitectura de *software* que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos.

P

Patrones GRASP: describen los principales fundamentos del diseño de objetos y la asignación de responsabilidades, expresados como patrones.

Patrones GOF: describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros.

U

UML (*Unified Modeling Language,* Lenguaje Unificado de Modelado): lenguaje de modelado de sistemas de *software*.