UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Sistema para el control de uso de los medios tecnológicos de los laboratorios docentes de la facultad 3.

Trabajo de diploma para optar por el Título de Ingeniero en Ciencias Informáticas.

Autor:
Juan Ignacio Morales Pestana

Tutores:
Ing. Zénel Reyes Pérez
Ing. Manuel Ramón Almaguer Ochoa

Declaración de autoría.

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

Para que así conste se firma la presente a los __ días del mes de _____ del año 2014.

Juan Ignacio Morales Pestana

_____

Firma de autor

Ing. Zénel Reyes Pérez

_____

Firma de tutor

Ing. Manuel Ramón Almaguer Ochoa

_____

Firma de tutor

Si no quieres perderte en el olvido tan pronto como hayas muerto, escribe cosas dignas de leerse o haz cosas dignas de escribirse.

---Alejandro Dumas (padre) ---

Dedico este trabajo a mi madre Teresa que siempre me ha apoyado y ha sido todo para mí, a Yolanda que ha sido mi segunda madre, a Marcos que es un padre para mí, a mis hermanos, a mi padre Juan Ramón.

Dedicado especialmente a mi novia Tania, que me escogió y conservó para hacerme feliz.

A estas personas les debo lo que soy hoy en día, desde lo más profundo de mi ser les agradezco.

Resumen

El surgimiento de la informática y su introducción en la informatización de las sociedades ha supuesto un reto para las entidades que la han incluido en su quehacer cotidiano para conseguir sus objetivos. Los medios tecnológicos, específicamente la computadora, constituyen uno de los principales elementos dentro de la gestión tecnológica de una entidad. Una computadora es una herramienta de trabajo que, en función del uso que se le dé, influye en el cumplimiento de los objetivos de las organizaciones, y esa es una de las razones por la que dichas organizaciones deben mantener control sobre estas.

En la actualidad son diversos los sistemas informáticos que permiten controlar el hardware y software de las computadoras, sin embargo, en algunos casos, dentro de sus funcionalidades no está contemplada cómo obtener el uso que se le da al medio tecnológico en su entorno y cómo influye el mismo en la degradación del medio y en función de estos, tomar decisiones.

El presente trabajo muestra el proceso de desarrollo de una solución informática, guiada por la metodología Feature Driven Development, que permite conocer el uso que se le da a determinado medio tecnológico basado en el tiempo que estuvo siendo utilizado por una persona, las aplicaciones ejecutadas en el mismo y el tipo de cierre realizado por la persona. Además se exponen las tecnologías, herramientas y materiales que se utilizaron para darle solución al problema presentado.

Palabras claves: aplicaciones ejecutadas, control de uso, medio tecnológico, tipo de cierre.

Tabla de contenidos.

Índice de tablas.

Índice de ilustraciones.

INTRODUCCIÓN

En la informatización de los procesos de las organizaciones, la introducción de las computadoras se ha convertido en una herramienta de apoyo para alcanzar los objetivos planificados. Debido a la importancia de las computadoras las organizaciones dedican recursos no sólo para adquirirlas sino también para su cuidado y mantenimiento, en este empeño resulta importante la gestión tecnológica.

La gestión tecnológica se define como la disciplina en la que se mezclan conocimientos de ingeniería, ciencias y administración con el fin de realizar la planeación, el desarrollo y la implantación de soluciones tecnológicas que contribuyan al logro de objetivos estratégicos y técnicos de una organización. La misma, entre otras, se compone de las siguientes actividades[1]:

- Seguimiento, análisis y prospectiva tecnológica.
- Identificación, evaluación y selección de nuevas tecnologías.
- Adaptación e innovación tecnológica.

Según van creciendo las organizaciones se necesitan herramientas para organizar y controlar los medios tecnológicos con que cuentan las mismas; con el propósito de mantener controlado todo el software y hardware instalado y usado en cada terminal de trabajo. En esta actividad las organizaciones se valen de herramientas para la gestión tecnológica, herramientas que cubren parcialmente las actividades mencionadas anteriormente, algunas de ellas detectan los usuarios con permisos en las computadoras, otras llegan un poco más lejos, brindando no sólo información de las aplicaciones instaladas sino también de las licencias relacionadas a las mismas.

Sin embargo, un elemento que no es considerado en algunos sistemas de control de hardware y software es el uso de los recursos y cómo influye esta variable en la depreciación del medio, incluso en el cumplimiento de los objetivos de la organización.

La Universidad de las Ciencias Informáticas (UCI) en su estructura está compuesta por facultades y estas, a su vez, por departamentos. Uno de los departamentos de los cuales se compone la Facultad 3 es el Departamento de Tecnología, cuya función principal es administrar los recursos tecnológicos de las áreas de la facultad, en especial el área docente.

Dicha administración cuenta con los siguientes elementos:

- La identificación [1]de los medios tecnológicos de las áreas[2] de la facultad.
- La ubicación de cada uno de los medios, la cual debe estar en correspondencia con la locación definida por el Departamento de Economía y Contabilidad.
- Control del uso que se hace de los medios.

Para llevar a cabo el control de los recursos tecnológicos y obtener los resultados descritos anteriormente, los especialistas del Departamento de Tecnología se valen de mecanismos que van desde la instalación de sistemas operativos personalizados, como son las imágenes docentes, hasta el uso de aplicaciones informáticas, papeles impresos y planillas de registro[3].

El Departamento de Tecnología en aras de promover la productividad del personal que se desempeña en la Facultad 3 ha puesto a disposición de la misma un sistema operativo personalizado que cuenta con un grupo de aplicaciones, las cuales en su mayoría son de interés institucional. La imagen docente en los últimos meses ha aumentado su capacidad debido al aumento del pedido de nuevas aplicaciones, elemento que ralentiza el proceso de despliegue en las distintas áreas docentes interrumpiendo en ocasiones la jornada laboral. Todo esto conlleva a que existan determinadas aplicaciones que no están siendo utilizadas o que presenten un bajo nivel de uso por parte de los usuarios, además del uso de aplicaciones para el ocio.

Para garantizar el control de uso de las terminales de trabajo en el área docente de la facultad 3, el Departamento de Tecnología se vale de un mecanismo que consiste en el registro de los datos de cada una de las personas que asisten a hacer uso de las computadoras, haciendo coincidir dichos datos personales con el identificador del medio[4] en el local correspondiente. Este proceso se realiza manualmente y es, en ocasiones, no supervisado, lo que puede traer consigo las siguientes deficiencias:

- No se registran los datos de las personas.

[1] Número otorgado por el Dpto. de Activos Fijos perteneciente a la Dirección de Contabilidad y Finanzas.
[2] Aulas, Laboratorios Docentes, Departamentos Docentes y Oficinas.
[3] Planilla utilizada para registrar los datos del personal que asiste a los laboratorios en horario no docente.
[4] Número de medio básico o nombre del equipo.

- No se registran los cambios de uso de las computadoras por parte de la misma persona.
- Existen inconsistencias o no registro de los horarios de entrada-salida del local.

Debido a las deficiencias que existen, los datos capturados en el proceso de control de uso no son totalmente verídicos, razón por la que no pueden ser usados para esclarecer ciertas y determinadas situaciones excepcionales.

El conjunto de información necesaria para llevar a cabo la gestión tecnológica en el Departamento de Tecnología es provista por diversas fuentes y es, en ocasiones, no confiable, además se encuentra sujeta a los límites de recursos con los que cuenta el departamento; lo cual dificulta el control sobre el uso de los medios tecnológicos.

De la situación planteada anteriormente se identificó el siguiente **problema:** ¿Cómo controlar el uso de los medios tecnológicos de los laboratorios docentes en la facultad 3?

La presente investigación tiene como **objeto de estudio:** La Gestión Tecnológica y se enmarca en el **campo de acción:** Los sistemas de control de medios tecnológicos.

Para dar solución al problema identificado se propone como o**bjetivo general:** Desarrollar un sistema informático que permita controlar el uso de los medios tecnológicos de los laboratorios docentes en la facultad 3.

El objetivo general quedaría desglosado en los siguientes **objetivos específicos:**

1. Fundamentar la investigación mediante la elaboración del Marco Teórico para sustentar los conceptos y la propuesta de solución.
2. Describir el sistema informático para controlar el uso de los medios tecnológicos de los laboratorios docentes.
3. Desarrollar el sistema informático para controlar el uso de los medios tecnológicos de los laboratorios docentes.
4. Validar técnicamente la solución propuesta mediante pruebas unitarias, pruebas de desempeño y pruebas de aceptación.

Para la investigación se plantea como **hipótesis:**

Si se desarrolla un sistema informático que permita conocer los usuarios que hacen uso de los medios tecnológicos y las aplicaciones instaladas en la imagen docente que se ejecutan en los mismos, entonces se podrá controlar el uso de los medios tecnológicos en los laboratorios docentes de la facultad 3.

Como **métodos investigativos** se utilizaron:

Teóricos:

- **Analítico – sintético:** permitió procesar a fondo el flujo de información del Departamento de Tecnología de la facultad 3, posibilitando así el procesamiento y diferenciación de la misma, enfocada hacia la investigación, permitiendo organizar y simplificar el análisis de todo el volumen de información.

Empíricos:

- **Observación:** este método permitió conocer a fondo los procesos realizados por el Departamento de Tecnología con el fin de identificar, entender y reproducir cada uno de sus elementos.
- **Entrevista:** con la aplicación de este método se esclarecieron los procesos realizados por el Departamento de Tecnología así como sus objetivos para el control de uso.

El presente trabajo se encuentra estructurado de la siguiente forma:

Capítulo 1. FUNDAMENTACIÓN TEÓRICA: Se presentan elementos relacionados con la ingeniería de software que permiten seleccionar la metodología para guiar el proceso de desarrollo del sistema informático y los artefactos necesarios durante el proceso. Se realiza un análisis de las herramientas que permiten el control de los medios tecnológicos y se proponen las herramientas, lenguajes y tecnologías para el desarrollo del sistema.

Capítulo 2. SISTEMA PARA EL CONTROL DE USO: Se describen las principales características que presenta el sistema a desarrollar, los principales componentes del mismo y su funcionamiento.

Capítulo 3. VALIDACIÓN DE LA SOLUCIÓN: Se describen las pruebas realizadas al sistema para su validación además de una muestra comparativa que refleja la situación del control de uso de los medios tecnológicos antes y después de los resultados obtenidos.

Capítulo 1: FUNDAMENTACIÓN TEÓRICA

1.1 Medios tecnológicos

Según el Diccionario de la Real Academia Española (DRAE) el término recurso se define como[2]:

- Medio de cualquier clase que, en caso de necesidad, sirve para conseguir lo que se pretende.
- Conjunto de elementos disponibles para resolver una necesidad o llevar a cabo una empresa.

Por otra parte se define también en el DRAE a la tecnología como el conjunto de teorías y de técnicas que permiten el aprovechamiento práctico del conocimiento científico[3]. Un recurso tecnológico, por lo tanto, es un recurso que se vale de la tecnología para cumplir con su propósito.

Muy similar a las definiciones expuestas de recurso se encuentra también una de las definiciones de medio, la cual plantea que un medio es algo que puede servir para un determinado fin[4].

Tomando en cuenta las similitudes existentes entre los términos recurso y medio mencionados, se define como medio tecnológico a todo recurso o medio que se valga de la tecnología para cumplir con su propósito.

1.2 Control de uso

El término control, según el DRAE, es comprobación, inspección, regulación manual o automática sobre un sistema[5]. De la misma manera define el término uso como acción y efecto de usar, ejercicio o práctica general de algo[6].

Para la presente investigación, se tomará como control de uso, al proceso de comprobar e inspeccionar las prácticas realizadas por los usuarios en los medios tecnológicos. El control de uso tendrá en cuenta los siguientes elementos:

- Tiempo que el medio estuvo siendo usado por una persona.

  La sesión de usuario constará de un inicio y un fin; los mismos estarán determinados por la hora y fecha en que el usuario se autentica en el medio y por la hora y fecha en que cierra su sesión.

- Aplicaciones de la imagen docente utilizadas en el medio.

  Durante el período de tiempo que una persona permanezca haciendo uso del medio, el mismo ejecutará un grupo de aplicaciones las que serán registradas.

- Tipo de cierre que realizó el usuario.

  Al finalizar, la persona tiene dos vías para cerrar su sesión, una de las vías es el cierre normal que consiste en el cierre de la sesión o apagado de la máquina. La otra vía existente es el cierre forzoso que consiste en el apagado forzoso [5]del medio.

1.3 Sistemas para el control de los medios tecnológicos

Las herramientas de control de medios tecnológicos cuentan, cada una, con disímiles características que son decisivas en el momento de incorporarlas a los procesos de una organización determinada, ya que las mismas deben cumplir con las políticas de la organización en cuestión. A continuación se enumeran los indicadores que se tendrán en cuenta en el estudio de los sistemas de control de los medios tecnológicos:

1. La identificación de los medios tecnológicos.
2. La ubicación física de cada uno de los medios.
3. El control de uso de los medios tecnológicos.
4. Licencia de software.
5. Ambiente de desarrollo.

1.3.1 Open Computers and Software Inventory Next Generation (OCS Inventory NG)

OCS Inventory NG es una herramienta que facilita el seguimiento de la configuración y el software instalado en los ordenadores de una red local, así como la instalación remota de aplicaciones desde un servidor web. Como ventaja es software GPL[6] y Open Source. También permite el despliegue de paquetes en sistemas operativos Windows y Linux. La herramienta se basa en estándares vigentes. Los diálogos

[5] Desconexión o reinicio forzoso.
[6] Del inglés GNU General Public Lisence. Garantiza a los usuarios finales la libertad de usar, estudiar, compartir y modificar el software.

entre los equipos clientes y el servidor se basan en HTTP [7] y el formato de los datos se realiza en XML[8][7].

1.3.2 Network Inventory Advisor (NIA)

NIA proporciona una solución que permite gestionar y auditar la red fácilmente. NIA es una herramienta de administración de red desarrollada en sus principios para el SO Windows y actualmente disponible también para MAC. La herramienta recopila todos los datos relevantes sobre los dispositivos de red para obtener un informe detallado de las estadísticas de hardware.[8]

NIA permite recoger información sobre los equipos de la red, realizar auditorías de software instalado y realizar un seguimiento de los cambios relacionados. El programa es capaz de escanear en Windows, Linux y Mac PC. También cuenta con un asistente de creación de informe para una creación de informes personalizados y plantillas de informes predefinidos[8].

Con la herramienta es posible agrupar, reagrupar y desagrupar, reasignar y editar múltiples nodos al mismo tiempo; realizar un seguimiento de los cambios de hardware y software cada vez que se ejecute un análisis nuevo.

1.3.3 Aranda ASSET Management (AAM)

Aranda ASSET Management (AAM) es una herramienta especializada en la gestión integral de inventarios actualizados de hardware y software, que permite tener el control del licenciamiento de software de todas las estaciones de trabajo, medir los niveles de uso de los programas en su organización y tomar el control remoto de las estaciones de trabajo minimizando los tiempos de desplazamiento de los especialistas de soporte[9].

La herramienta ofrece los siguientes beneficios[9]:

- Conocimiento real de los recursos de la organización.
- Inventario actualizado y detallado de hardware, software y dispositivos asociados a cada estación.

[7] Del inglés HyperText Transfer Protocol.
[8] Del inglés eXtensive Markup Language.

- Control permanente del uso de los recursos informáticos por estación.
- Obtener la información de los procesos en tiempo real y mejorar la toma de decisiones.
- Disminución en el robo o pérdida de partes.
- Medir y auditar el uso de software en cada estación.

1.3.4 Gestor de Recursos de Hardware y Software

El Gestor de Recursos de Hardware y Software (GRHS) es un sistema informático basado en la arquitectura cliente-servidor con el objetivo de realizar el inventario de hardware y software en una red de computadoras[10]. Esta solución fue desarrollada en el Centro de Telemática de la Universidad de las Ciencias Informáticas, Cuba.

La arquitectura de GRHS es N-Tiers y aplica ingeniería basada en componentes para las aplicaciones. Está compuesta por tres aplicaciones: gclient, gserver y gadmin. La aplicación gclient se encarga de recopilar la información de los clientes, detectar los cambios, detectar las incidencias y tomar las acciones de control. La aplicación gserver recopila la información de todos los inventarios, almacena las configuraciones y tiene la información de todos los clientes. La aplicación gadmin es la consola de administración de GRHS. Esta es utilizada para establecer las configuraciones, enviar órdenes a los clientes, mostrar reportes de los inventarios y mostrar la información de los clientes[10].

1.3.5 Sistema de control de acceso del Centro de Identidad y Seguridad Digital (SCACISED)

La herramienta es desarrollada en el Centro de Identidad y Seguridad Digital de la Universidad de las Ciencias Informáticas. La misma hace uso de herramientas como Nagios que le permiten mantener un control del uso de los servicios por las diferentes terminales de trabajo. Los servicios controlados son el correo, DNS[9], Proxy, bases de datos y servidor de aplicaciones, entre otros. SCACISED haciendo uso de otras herramientas como el OCS Inventory y Sense establece una organización física de las áreas y ubicación lógica de las redes lo que le permite monitorear la actividad de la red y los servicios que fluyen a través de la misma[11].

[9] Del inglés Domain Name System.

1.3.6 Resultado del estudio de herramientas para el control de los medios tecnológicos

Las soluciones informáticas para el control de los medios tecnológicos que han sido descritas en esta investigación cubren parcialmente los objetivos del Departamento de Tecnología para el control de uso. En la siguiente tabla se muestra el resultado de la evaluación de las herramientas donde la (**x**) significa que la herramienta cumple con el indicador, en caso contrario se usará (**-**)**.**

Tabla 1: Resultado del estudio de las herramientas.

Indicadores/Herramientas		**OCS Inventory-NG**	**NIA**	**AAM**	**GRHS**	**SCACISED**
Identificación		(x)	(x)	(x)	(x)	(x)
Ubicación física		(-)	(x)	(x)	(x)	(x)
Control de uso	Tiempo	(-)	(-)	(-)	(-)	(x)
	Aplicaciones ejecutadas	(-)	(-)	(x)	(-)	(-)
	Tipo de cierre	(-)	(-)	(-)	(-)	(-)
Software libre		(x)	(-)	(-)	(x)	(x)
Ambiente de desarrollo		PHP	.NET	.NET	Python	PHP
Total		**2**	**2**	**3**	**3**	**4**

En la tabla anterior se evidencia que el OCS Inventory no cumple con la mayoría de los objetivos planteados para el control de uso, no siendo así el caso de las herramientas NIA y AAM que cumplen parcialmente los objetivos pero son soluciones privativas y no se puede tener acceso a su código fuente lo cual no está acorde a la política adoptada por nuestro país de soberanía tecnológica.

En cuanto a GRHS no cumple con los elementos de control de uso y además su ambiente de desarrollo no cumple con las herramientas establecidas en la propuesta de informatización de la Facultad 3[13]. Por otro lado la herramienta actualmente se encuentra en las fases de desarrollo y prueba; aún no ha sido desplegada y según su cronograma no estará integrada con el sistema operativo Nova, instalado en los laboratorios docentes, hasta diciembre del 2014.

SCACISED no cumple con algunos de los indicadores establecidos, la solución posee en su entorno de aplicaciones cinco aplicaciones para las cuales es necesario cierto ambiente de despliegue y capacitación por parte de los usuarios del sistema, además de necesitar de un recurso humano para la autorización y

asignación de los permisos en los medios. La herramienta se encuentra en las fases de desarrollo y pruebas.

De lo anteriormente expuesto se concluye:

- Las herramientas capaces de brindar información referente al uso de los medios tecnológicos no pueden ser utilizadas debido a que las mismas no cumplen con las el ambiente de desarrollo para los sistemas de la Facultad 3 o las condiciones para su despliegue no se encuentran creadas.
- Debido a la importancia de conocer la información referente al uso de los medios tecnológicos y que la misma, por razones ya expuestas, no puede ser completamente brindada por las herramientas descritas; se hace necesaria la implementación de un sistema capaz de garantizar dicha información.
- La solución debe ser una aplicación web según lo establecido en la propuesta de informatización de la Facultad 3.[12]
- Para el desarrollo de la solución se debe hacer uso de tecnologías y herramientas libres.

1.4 Metodología, lenguajes y herramientas

1.4.1 Proceso de desarrollo de software

Un proceso, es un grupo de actividades, acciones y tareas que son ejecutadas cuando un producto está por ser creado[13]. Un proceso define quién está haciendo qué, cuándo y cómo para alcanzar cierta meta[14].

El proceso de desarrollo de software, es aquel en que las necesidades del usuario son traducidas en requerimientos de software, los que son transformados en diseño y este implementado en código; es probado, documentado y certificado para su uso operativo[15].

En el proceso de desarrollo de software resulta necesario que los proyectos tengan un fin exitoso con óptima calidad y que satisfagan las necesidades del cliente; para alcanzar estos objetivos, se usan diferentes metodologías o conjunto de procedimientos, técnicas, herramientas y soportes documentales que ayudan a los ejecutores a realizar el nuevo software.

El éxito del producto depende en gran medida de la metodología escogida por el equipo, ya sea esta tradicional o ágil. El equipo debe seleccionar una metodología adecuada que le permita explotar al máximo su potencial y aumentar la calidad del producto con los recursos y los tiempos establecidos.

Teniendo en cuenta las características de la solución que se desea desarrollar y el equipo de trabajo, los modelos ágiles, junto a los modelos de prototipos, proveen un marco idóneo para el desarrollo del producto, debido a que este tipo de modelos ofrece las siguientes ventajas:

- Ahorro de tiempo y recursos.
- Movimiento rápido, los avances más recientes pueden ser rápidamente codificados y probados usando este método.
- Los errores pueden ser fácilmente corregidos.
- Es un método controlado dinámicamente, que insiste en la actualización frecuente de los progresos en el trabajo a través de reuniones regulares.
- Se requiere constante retroalimentación del cliente.
- La retroalimentación constante, facilita los cambios.
- Los chequeos son realizados con frecuencia, lo que hace posible medir la productividad individual, esto conduce a la mejora del rendimiento de cada uno de los miembros del equipo.
- Los problemas se identifican con suficiente antelación a través de las reuniones frecuentes del equipo y por lo tanto se pueden resolver con rapidez.

Dentro de las metodologías ágiles se encuentran[16]:

Extreme Programing (XP): Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. XP se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y facilidad para enfrentar los cambios. XP se

define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes donde existe un alto riesgo técnico.

SCRUM: Define un marco para la gestión de proyectos y se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos:

- El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente.
- La segunda característica importante son las reuniones a lo largo del proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración.

Feature-Driven Development (FDD): Define un proceso iterativo que consta de cinco pasos con iteraciones cortas. Define pasos como el desarrollo de un modelo general, construcción de una lista de funcionalidades, planificación en base a las funcionalidades a implementar y posteriormente el diseño y la implementación de las funcionalidades identificadas. Se centra en las fases de diseño e implementación del sistema partiendo de una lista de características que debe reunir el software siendo las dos últimas fases las de más larga duración durante el proceso de desarrollo. Sus impulsores son Jeff De Luca [10]y Peter Coad[11].

Para el desarrollo de la solución se ha seleccionado FDD, observando que una de las principales ventajas de esta metodología de desarrollo de software es que permite partir de un análisis previamente realizado. FDD no hace énfasis en la especificación de requerimientos sino que se enfoca en el diseño y la construcción de la solución a partir de un listado de funcionalidades identificadas. Sin embargo, puede trabajar con otras metodologías de desarrollo de software y no requiere la utilización de ningún modelo de proceso específico, además, hace énfasis en aspectos de calidad durante todo el proceso e incluye un monitoreo permanente del avance del proyecto. También FDD ofrece mayor predictibilidad si los requerimientos del proyecto se mantienen estables y enfoca sus esfuerzos en la primera fase del ciclo de vida llamada *"Proceso número uno"* la cual es la encargada de comprender el recopilar los requisitos del

[10] Ejecutivo y estratega dinámico de Tecnologías de la Información.
[11] Ms. Ciencias de la Computación y Bachiller con honores en Ingeniería Eléctrica.

cliente[17]. Esta metodología opta por la práctica de código propietario, argumentando que cada uno de los desarrolladores conoce mejor su propio código y la consecuencia de los cambios en el mismo[17][18].

1.4.2 Lenguajes

Lenguaje Unificado de Modelado (UML[12])

UML es un leguaje de modelado visual estandarizado de propósito general en el campo de la ingeniería de software orientada a objetos. La norma fue creada por el Object Management Group (OMG) en 1997 y desde entonces se ha convertido en el estándar del sector para el modelado de sistemas de software. UML incluye un conjunto de técnicas de notación gráfica para crear modelos visuales de programación orientada a objetos[13][19].

UML se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios[14].

La versión seleccionada para el diseño haciendo uso de UML es la número ocho.

PHP

PHP es un lenguaje de programación de uso general, de código del lado del servidor, originalmente diseñado para el desarrollo web de contenido dinámico. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página Web resultante. PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes, y además, puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo[20].

Características:[20]

- Orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.

[12] Del inglés Unified Modeling Language.

- Es considerado un lenguaje fácil de aprender, ya que en su desarrollo se simplificaron distintas especificaciones, como es el caso de la definición de las variables primitivas, ejemplo que se hace evidente en el uso de php arrays.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- Capacidad de expandir su potencial utilizando módulos (llamados ext's o extensiones).
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de programación orientada a objetos. Incluso aplicaciones como Zend framework y Symfony, están totalmente desarrolladas mediante esta metodología.
- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- Tiene manejo de excepciones (desde PHP5).

La versión del lenguaje a utilizar será la 5.3.

Bourne Again SHell (BASH)

BASH es un shell Unix libre que se puede utilizar en lugar del shell Bourne. Se trata de una aplicación completa de la Interfaz IEEE de Sistema Operativo Portable para Unix (POSIX)[13].

BASH es básicamente un procesador de comandos que normalmente se ejecuta en una ventana de texto, lo que permite al usuario escribir manualmente los comandos que causan acciones, puede leer comandos de un archivo, llamado script, y además, al igual que todos los shells de Unix soporta lo siguiente[21]:

- Comodines
- Tuberías.
- Ejecución de comandos.

[13] Del inglés Portable Operating System Interface for UNIX.

- Variables y estructuras de control para pruebas condicionales e iteraciones.

El objetivo principal de BASH es permitir al usuario interactuar con el sistema operativo; usualmente involucra la ejecución de programas y comandos de textos tecleados en la pantalla. BASH permite también la automatización de tareas para su uso reproductivo.[22]

1.4.3 Frameworks y librerías de desarrollo

Symfony

Symfony es un marco de trabajo para el desarrollo de aplicaciones en el lenguaje PHP. Este marco de trabajo permite la utilización de componentes genéricos, evitando la implementación de tareas comunes y enfocando a los desarrolladores en los verdaderos desafíos de la aplicación.[23]

Symfony2 introduce el concepto de bundle, este concepto se refiere a un conjunto de archivos (PHP, hojas de estilo, JavaScript, o cualquier otro) que implementan una funcionalidad específica.

Un bundle es el homólogo de un plug-in en otro software, pero con la diferencia de que todo en Symfony2 es un bundle, incluyendo el código del marco de trabajo así como el de la aplicación desarrollada. Esto permite la utilización de funciones empaquetadas dentro de bundles de terceros y la distribución de propios, así como la selección de las prestaciones a habilitar en la aplicación para optimizarla en la manera deseada[24].

Dentro de sus características se pueden citar[25]:

- **Rápido y menos abarcador:** Symfony fue concebido desde el principio para ser rápido y favorecer el rendimiento.
- **Gran flexibilidad:** Su inyección de dependencia y el despachador de eventos hacen que sea totalmente configurable con cada una de las partes que son totalmente independientes.
- **Expandible:** Desde el componente más pequeño hasta el propio núcleo del framework, todo se presenta como un bundle. Cada paquete está diseñado para añadir funcionalidad al framework. En cualquier caso, el sistema de bundles permite que todo cambie en Symfony, incluyendo el propio

núcleo. El comportamiento de la estructura de este modo se puede cambiar a voluntad, sin necesidad de reconfiguración completa.

- **Estable y sostenible:** Desarrollado por SensioLabs, las principales versiones de Symfony son soportados por 3 años por la empresa, e incluso para la vida en lo que se refiere a cuestiones relacionadas con la seguridad.
- **Confort al desarrollar:** Como un entorno altamente funcional, Symfony también garantiza un cierto nivel de comodidad para los desarrolladores
- **Facilidad de uso:** Totalmente flexible para satisfacer las necesidades de los profesionales y usuarios avanzados por igual, Symfony también es muy accesible, cuenta con abundante documentación, una gran comunidad y el apoyo profesional.

Por las características analizadas anteriormente Symfony en su versión 2.4.1 será utilizado como marco de trabajo.

Object Relational Mapper (ORM) Doctrine

Un ORM es una técnica de programación que permite convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, es decir, las tablas de nuestra base de datos pasan a ser clases y los registros, objetos que podemos manejar con facilidad[26].

Doctrine es un ORM para PHP 5.3 o superior que proporciona persistencia transparente a objetos PHP, se sitúa en una capa de abstracción de base de datos para realizar la transformación de las tablas en objetos PHP utilizando un dialecto SQL propio orientado a objeto llamado DQL.[27]

Una característica de Doctrine es el bajo nivel de configuración que necesita para empezar un proyecto. Doctrine puede generar clases a partir de una base de datos existente y después el programador puede especificar relaciones y añadir funcionalidad extra a las clases autogeneradas. No es necesario generar o mantener complejos esquemas XML de base de datos como en otros frameworks[28].

La versión de Doctrine ORM seleccionada para el desarrollo de la solución es la versión 2.0.

Twitter Bootstrap

Bootstrap es una colección de herramientas de software libre para la creación de sitios y aplicaciones web. Fue desarrollado por Mark Otto [14]y Jacbod Thornton [15]de Twitter, como un framework para fomentar la consistencia a través de herramientas internas. Antes de Bootstrap, se usaban varias librerías para el desarrollo de interfaces de usuario, las cuales guiaban a inconsistencias y a una carga de trabajo alta en su mantenimiento[29].

Bootstrap tiene un soporte relativamente incompleto para HTML5 y CSS3, pero es compatible con la mayoría de los navegadores web. La información básica de compatibilidad de sitios web o aplicaciones está disponible para todos los dispositivos y navegadores[29].

Desde la versión 2.0 también soporta diseños sensibles. Esto significa que el diseño gráfico de la página se ajusta dinámicamente, tomando en cuenta las características del dispositivo usado (Computadoras, tabletas, teléfonos móviles). Bootstrap es de código abierto[29].

La versión seleccionada para el desarrollo de la solución será la 3.1.1.

jQuery

jQuery es una biblioteca gratuita de Javascript, cuyo objetivo principal es simplificar las tareas de creación de páginas web responsivas, acorde a lo estipulado en la Web 2.0, la cual funciona en todos los navegadores modernos. Por otro lado, jQuery ayuda a que el desarrollador se enfoque en el diseño del sitio, al abstraer por completo todas las características específicas de cada uno de los navegadores, otra de las grandes ventajas de jQuery es que se enfoca en simplificar los scripts y en acceder/modificar el contenido de una página web, y además, jQuery agrega una cantidad impresionante de efectos nuevos a Javascript, los cuales podrán ser utilizados en los sitios Web[30].

Beneficios del uso de jQuery[30]:

- jQuery utiliza sintaxis muy parecida a CSS.
- Funciona con series de elementos.

[14] Diseñador de GitHub, previo diseñador de Twitter
[15] Diseñador de Twitter.

- Compatible con todos los navegadores modernos.

La última versión estable de jQuery es la 1.10.2, la cual es seleccionada para ser utilizada en el desarrollo de la solución informática propuesta junto con la versión 1.10.4 de jQuery-UI.

1.4.4 Herramientas

Visual Paradigm

Visual Paradigm es una herramienta de ingeniería de software asistida por computación (CASE)[16], la misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación[31]. Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta de software libre de probada utilidad para el analista que fue diseñada para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos.

Entre sus principales características se pueden mencionar[31]:

- Entorno de creación de diagramas para UML.
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales IDEs[17].

[16] Del inglés Computer Asisted Software Engineering.
[17] Del inglés Integrated Development Environment. Entorno de desarrollo integrado.

- Generación de bases de datos.
- Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Importación y exportación de ficheros XML.

NetBeans

NetBeans es un entorno de desarrollo integrado (IDE)[18]de código abierto para desarrolladores de software. Cuenta con todas las herramientas necesarias para crear aplicaciones profesionales de escritorio, empresariales, web y aplicaciones móviles con la plataforma Java, así como con C / C + +, PHP, JavaScript y Groovy[32].

El editor de PHP de NetBeans ofrece plantillas de código y generación del mismo (métodos *get* y *set*), la refactorización, información sobre herramientas de parámetros, consejos y soluciones rápidas, y la finalización de código inteligente[33].

El IDE también ofrece un editor de HTML, JavaScript y CSS, con el cual provee el resaltado de sintaxis, completamiento de código, y la comprobación de errores para HTML, CSS y JavaScript. El editor reconoce código HTML en los archivos de JavaScript y viceversa y reconoce también HTML y JavaScript en XHTML, PHP y archivos JSP[33].

NetBeans posee soporte para el marco de trabajo Symfony, una de las razones principales además de las expuestas por la que es seleccionado como entorno de desarrollo. La herramienta se utilizará en su versión 7.3.

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD[19] y con su código fuente disponible libremente. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando[34].

[18] Del inglés Integrated Development Environment.
[19] Del inglés Berkeley Software Distribution. Es una licencia de software libre permisiva.

Su desarrollo comenzó hace más de 16 años, y durante este tiempo, estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema[35].

Entre las características fundamentales de PostgreSQL se pueden encontrar[35]:

- Excelente cumplimiento de los estándares SQL 2008.
- Arquitectura cliente-servidor.
- Diseño altamente concurrente donde los lectores y escritores no se bloquean entre sí.
- Altamente configurable y extensible para muchos tipos de aplicaciones.
- Integridad referencial (Foreign Keys).
- Tiene licencia de tipo BSD.

Para el desarrollo de la solución propuesta se hará uso de la versión 9.1 del SGBD.

1.5 Conclusiones del capítulo

- Este capítulo fue esencial para conceptualizar el control de uso de los medios tecnológicos de la Facultad 3, permitiendo tener un acercamiento inicial al dominio del problema identificado.
- El uso de indicadores para el estudio de las herramientas de control de medios tecnológicos esclareció el análisis de las mismas permitiendo identificar similitudes, diferencias y fortalezas tanto de las soluciones internacionales como las desarrolladas en nuestro país.
- Dentro de las similitudes de las herramientas estudiadas se encuentra que hacen uso, ya sea de manera opcional o requerida, de agentes para la recogida de información. Se precisaron las tecnologías, herramientas y la metodología para el desarrollo de la solución.

Capítulo 2: PROPUESTA DE SOLUCIÓN

Teniendo en cuenta los objetivos del presente trabajo y haciendo uso de la información expuesta en el capítulo anterior se propone como solución, un sistema informático soportado por tecnologías web que posibilite comprobar e inspeccionar el uso que se le da a los medios tecnológicos de los laboratorios docentes de la Facultad 3 de la Universidad de las Ciencias Informáticas.

2.1 Descripción de la propuesta de solución

La solución, en primera instancia, debe ser capaz de recopilar toda la información del tiempo que se estuvo utilizando un medio tecnológico y las aplicaciones ejecutadas en los mismos, haciendo uso de agentes remotos que se encargan del envío de esta información para su posterior procesamiento por parte del sistema. El procesamiento de los ficheros enviados por los agentes se realizará de forma automática. Por otro lado el sistema deberá permitir adjuntar ficheros aislados de los medios tecnológicos que hayan presentado problemas, impidiendo que los agentes envíen la información en el momento programado.

Para garantizar la inspección y comprobación del uso de los medios tecnológicos el sistema permitirá realizar la búsqueda de aplicaciones y sesiones de usuario, atendiendo a diferentes criterios de selección los cuales son sugeridos para obtener el resultado deseado.

La herramienta debe permitir también la gestión manual de la información asociada, lo cual significa que se podrán gestionar los distintos medios tecnológicos, áreas de responsabilidad y tipos de medios tecnológicos. De la misma forma en que el sistema permitirá gestionar distintas entidades y otros parámetros configurables, contará con un grupo de elementos no configurables que son necesarios para el correcto funcionamiento del sistema y su cambio podría implicar un mal funcionamiento del mismo.

En orden para garantizar el flujo de información hacia otras áreas el sistema brindará la posibilidad de generar reportes o resúmenes en formato PDF[20] de acuerdo a la información obtenida de las búsquedas

[20] Portable Document Format.

realizadas por el usuario. De la misma forma el sistema mostrará gráficas que resumen el uso de los medios tecnológicos permitiendo exportar las mismas en formato PNG.[21]

2.2 Modelo de dominio

Uno de los pasos esenciales de un análisis o investigación orientado a objetos es descomponer el problema en conceptos u objetos individuales. Un modelo de dominio es una representación de conceptos del dominio de un problema. En UML se define como un diagrama estático donde no se definen operaciones. En el diagrama de dominio se pueden mostrar[36]:

- Conceptos.
- Relaciones entre conceptos.
- Atributos de conceptos.

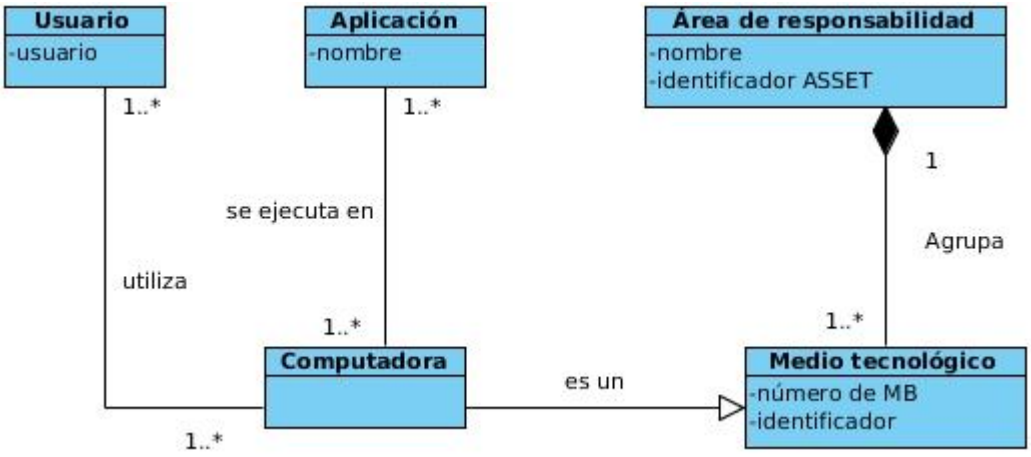

Ilustración 1: Modelo de dominio de la solución.

2.3 Requerimientos identificados.

Para realizar la implementación de la solución propuesta se partió de un análisis donde se identificaron los rasgos[22] con que debería contar el sistema, los mismos se encuentran divididos en cuatro bundles diferentes y definidos con la estructura <acción> <resultado> <objeto> según lo plantea la metodología seleccionada para guiar el proceso de desarrollo de software[37]. Para la obtención de estos rasgos se

[21] Portable Network Graphic.
[22] Denominación para los requerimientos según la metodología.

utilizaron las técnicas de tormenta de ideas y entrevista. A continuación se presentan los rasgos identificados y si el mismo se considera crítico[23] o no.

AdministraciónBundle.

Grupo de rasgo 1: Administrar un área de responsabilidad

Tabla 2: Rasgos de Administrar un área de Responsabilidad.

Número	Rasgo	Crítico	Código
1	Adicionar área de responsabilidad al sistema.	No	FS1R1
2	Modificar área de responsabilidad registrada en el sistema.	No	FS1R2
3	Buscar áreas de responsabilidad registradas en el sistema.	No	FS1R3
4	Eliminar área de responsabilidad registrada en el sistema.	No	FS1R4

Grupo de rasgo 2: Administrar un tipo de medio tecnológico

Tabla 3: Administrar un tipo de medio tecnológico.

Número	Rasgo	Crítico	Código
5	Adicionar tipo de medio tecnológico al sistema.	No	FS2R5
6	Modificar tipo de medio tecnológico registrado en el sistema.	No	FS2R6
7	Buscar tipos de medios tecnológicos registrados en el sistema.	No	FS2R7
8	Eliminar tipo de medio tecnológico registrado en el sistema.	No	FS2R8

Grupo de rasgo 3: Administrar un medio tecnológico

Tabla 4: Rasgos de Administrar un medio tecnológico.

Número	Rasgo	Crítico	Código
9	Adicionar medio tecnológico al sistema.	No	FS3R9
10	Modificar medio tecnológico registrado en el sistema.	No	FS3R10
11	Buscar medios tecnológicos registrados en el sistema.	No	FS3R11
12	Adicionar medios tecnológicos de un fichero csv.	No	FS3R12

[23]Se considera como rasgo crítico aquellos que recibirán grandes cantidades de peticiones por parte del usuario y manejan grandes cantidades de datos.

Grupo de rasgo 4: Configurar el sistema

Tabla 5: Rasgos de Configurar el sistema.

Número	Rasgo	Crítico	Código
13	Configurar la ruta de localización de ficheros.	No	FS4R13

ControlSesionesBundle.

Grupo de rasgo 5: Controlar las sesiones de usuario

Tabla 6: Rasgos de Controlar las sesiones de usuario.

Número	Rasgo	Crítico	Código
14	Registrar datos de sesiones de usuarios en el sistema.	No	FS5R14
15	Buscar sesiones de usuarios registradas en el sistema.	Sí	FS5R15
16	Adjuntar ficheros de sesiones aislados al directorio de sesiones.	No	FS5R16

ControlAplicacionesBundle

Grupo de rasgo 6: Controlar uso de aplicaciones

Tabla 7: Rasgos de Controlar uso de aplicaciones.

Número	Rasgo	Crítico	Código
17	Registrar datos de uso de aplicaciones en el sistema.	No	FS6R17
18	Adjuntar ficheros de aplicaciones aislados al directorio de aplicaciones.	No	FS6R18
19	Buscar uso de aplicaciones registradas en el sistema.	Sí	FS6R19

General

Grupo de rasgo 7: Generar una estadística

Tabla 8: Rasgos de Generar una estadística.

Número	Rasgo	Crítico	Código
20	Buscar resumen de uso de medios tecnológicos.	No	FS7R20
21	Mostrar estadísticas de medios tecnológicos más usados.	Sí	FS7R21
22	Mostrar estadística resumen de uso de las áreas de responsabilidad.	Sí	FS7R22
23	Mostrar estadísticas de usuarios más frecuentes.	Sí	FS7R23
24	Mostrar estadísticas de aplicaciones más usadas.	Sí	FS7R24

25	Buscar resumen de uso de aplicaciones.	No	FS7R25
26	Mostrar estadísticas de un medio tecnológico.	Sí	FS7R26
27	Buscar resumen de usuario.	No	FS7R27
28	Mostrar estadística resumen de usuario.	Sí	FS7R28

ServiciosBundle

Grupo de rasgo 8: Generar un reporte

Tabla 9: Rasgos de Generar un reporte.

Número	Rasgo	Crítico	Código
29	Exportar en formato PDF el resultado de la búsqueda de sesiones.	No	FS8R29
30	Exportar en formato PDF el resultado de la búsqueda de resúmenes de usuarios.	No	FS8R30
31	Exportar en formato PDF el resultado de la búsqueda de resúmenes de medios tecnológicos.	No	FS8R31
32	Exportar en formato PDF el resultado de la búsqueda de uso de aplicaciones.	No	FS8R32

2.1.2 Requerimientos no funcionales

Los requerimientos no funcionales son restricciones de los servicios o funciones que brinda el sistema y que no se refieren directamente a las funcionalidades del mismo sino a sus propiedades emergentes como los tiempos de respuesta y la capacidad de almacenamiento.[38]

Apariencia o interfaz externa

La interfaz del sistema debe adaptarse al entorno en que el usuario la visualiza.[39]

Seguridad

- **Autenticación:** Para que un usuario se pueda autenticar ante el sistema debe cumplir con los siguientes requisitos:
  - ✓ El usuario debe ser válido en el dominio UCI.
  - ✓ La contraseña debe ser ingresada correctamente según le corresponda al usuario.
  - ✓ El usuario debe estar registrado en la base de datos del sistema.

- **Autorización:** A cada uno de los usuarios se le asignarán uno o más roles y un grupo de acciones independientes en caso de ser necesario. Luego de que el usuario se autentique ante el sistema en la sesión creada por el marco de trabajo se cargarán cada una de las acciones a las que el usuario tiene permiso, de esta forma el sistema mostrará solamente las funcionalidades a las que el usuario puede acceder.

Rendimiento

Los tiempos de respuesta y velocidad de procesamiento de la información serán rápidos, no mayores de dos segundos para los registros, eliminaciones, modificaciones y búsquedas razonadas; en el caso de la búsqueda de resúmenes y de generación de reportes los tiempos de respuesta no deberán exceder los 20 segundos.

Portabilidad

El sistema debe ser multiplataforma con énfasis en la plataforma Linux.

Software

Del lado del cliente:

- Navegador Mozilla Firefox 24 o superior.
- Sistema operativo Windows 7 o superior o Linux en cualquiera de sus distribuciones.

2.4 Diseño del sistema.

2.4.1 Arquitectura

La arquitectura del sistema está compuesta por cuatro capas: Presentación, Controladora, Datos y una capa adicional de servicios para centralizar las funcionalidades comunes utilizadas por los bundles.

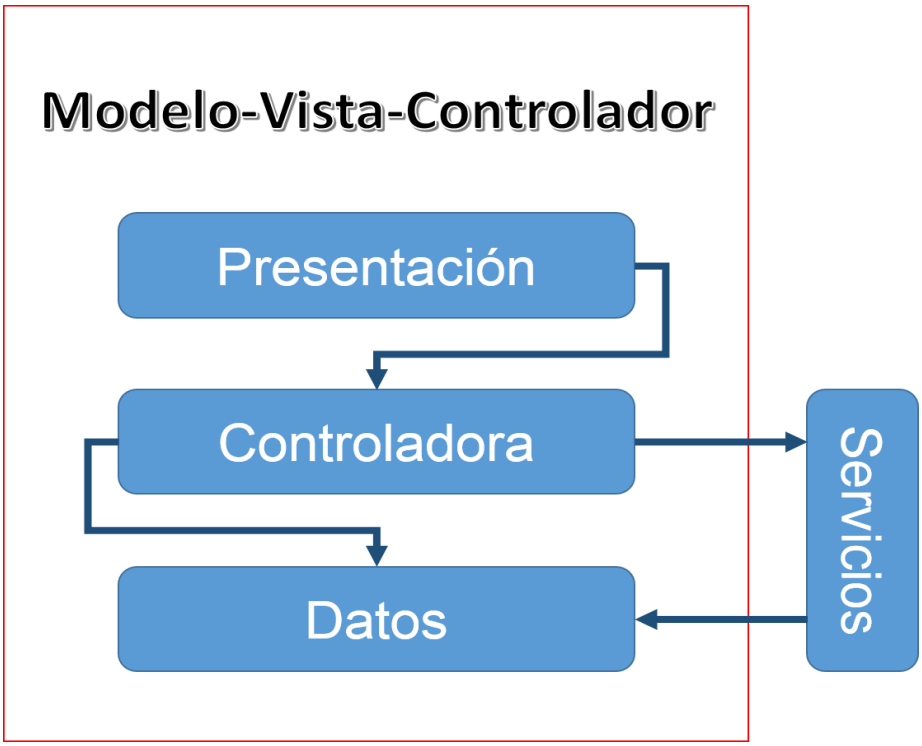

Ilustración 2: Arquitectura del sistema.

2.4.2 Patrones de diseño

Patrones generales de software para la asignación de responsabilidades (GRASP[24])

- **Creador:** En las clases controladoras se encuentran todas las acciones definidas para los distintos módulos o Bundles de la solución. En las acciones de cada una de estas clases controladoras se

[24] Del inglés General Responsability Assignment Software Patterns.

crean los objetos que representan las entidades, evidenciando de este modo que ellas son creadoras de dichas entidades.

- **Experto:** Symfony 2 utiliza el ORM Doctrine 2 para implementar la capa de abstracción en el modelo, encapsula toda la lógica de los datos y son generadas las clases con todas las funcionalidades de las entidades, demostrando así que el ORM Doctrine es el "experto" en el modelo de datos.
- **Controlador:** Todas las peticiones realizadas a la aplicación son manejadas por un único controlador frontal (app.php), que es el punto de entrada de toda la aplicación en un entorno determinado, administrando así todos los eventos del sistema. Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de la acción solicitada y el nombre de un bundle con la URL[25] solicitada por el cliente.
- **Bajo acoplamiento:** Las clases controladoras heredan solamente de una única clase Controller.php para lograr un bajo acoplamiento de las mismas.
- **Alta cohesión:** Symfony permite organizar la estructura del proyecto así como la asignación de responsabilidades con una alta cohesión. Ejemplo de esto son las clases controladoras, que son las encargadas de definir todas las acciones de las vistas así como colaborar con otras clases controladoras.

Banda de los cuatro (GoF26)

- **Inyección de dependencia:** Permite la eliminación de las dependencias incluidas en el código (use, require, etc), por lo que es posible cambiarlas, ya sea en tiempo de ejecución o en tiempo de compilación.
- **Decorador:** En Symfony el archivo base.html.twig contiene todo el código HTML que es común para todas las vistas del sistema, de igual forma contiene las hojas de estilos y archivos java script que definen un comportamiento uniforme el resto de las plantillas. La herencia puede ser aplicada en profundidad haciendo que las vistas de cada uno de los bundles hereden de una plantilla en específico que define un comportamiento y un estilo específico para dicho bundle.

[25] Del inglés Uniform Resource Locator.
[26] Del inglés Gand Of Four.

- **Inicialización tardía:** La inicialización tardía es la técnica de retrasar la creación de un objeto, el cálculo de un valor u otro proceso hasta la primera vez en que es realmente necesario[40]. Las clases AplicacionesService y SesionesService son definidas como servicios internos de la aplicación obteniendo la información cuando es realmente solicitada evitando así el consumo de memoria innecesario.

2.4.3 Diagrama de clases

Los diagramas de clases describen por lo menos el nombre de las clases, las superclases, las etiquetas de los métodos y los atributos simples de una clase. Esa información es suficiente para formular una definición básica de una clase en un lenguaje orientado a objetos[36].

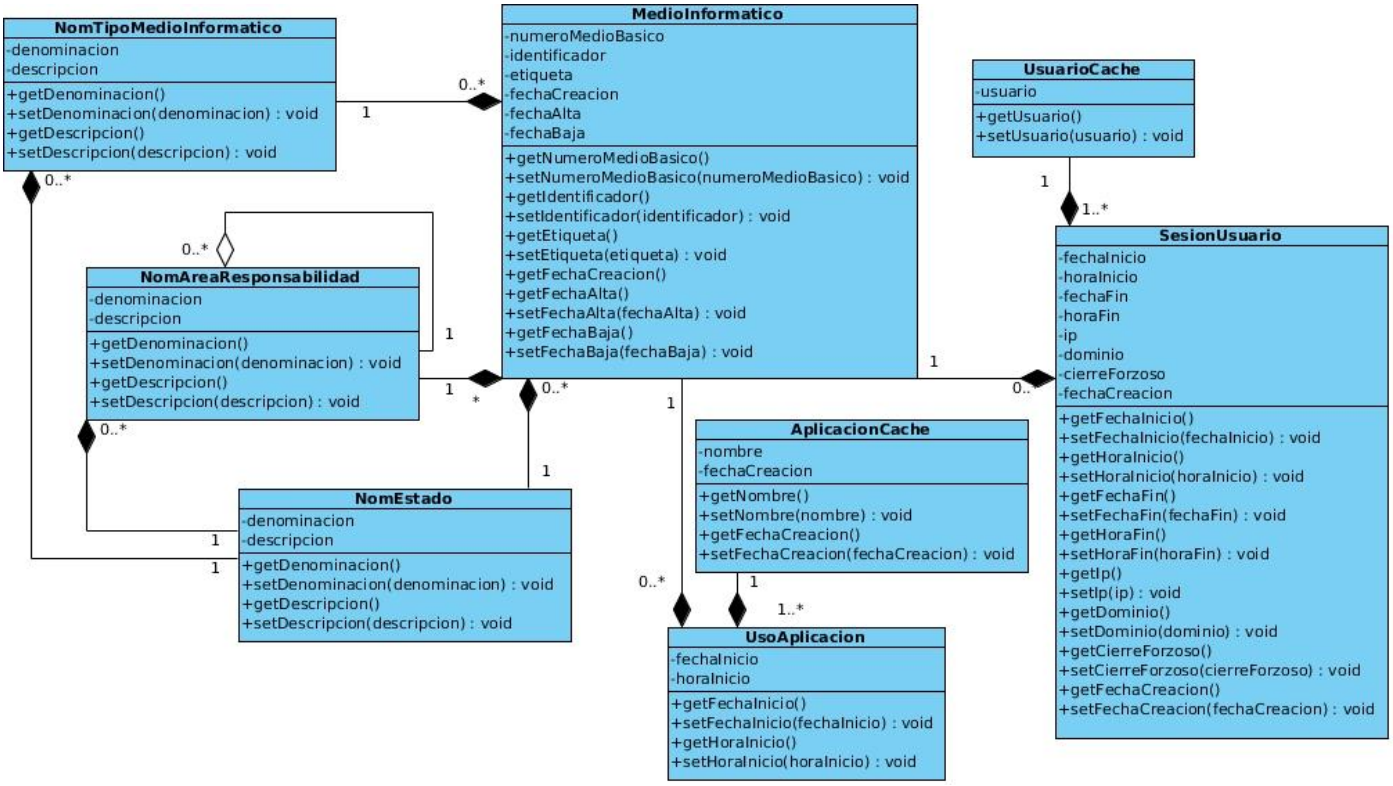

Ilustración 3: Diagrama de clases de la solución.

Descripción de las clases

Se identificaron ocho clases, a continuación se detallan dos de las mismas; el resto se encuentra disponible en el anexo ***DT-CEUS-Descripcion-Clases.doc***.

Tabla 10: Descripción de la clase NomAreaResponsabilidad.

Tipo de clase: Entidad	
Nombre: NomAreaResponsabilidad	**Propietario:** Juan Ignacio Morales Pestana
Atributo	**Tipo**
denominacion	String
descripcion	String
asset	String
idAreaResponsabilidadSuperior	NomAreaResponsabilidad
idEstado	NomEstado
Para cada rasgo asociado	
Nombre	Adicionar un área de responsabilidad al sistema.
Descripción	Valida los datos introducidos y registra la nueva área en caso de no existir errores.
Nombre	Modificar un área de responsabilidad registrada en el sistema.
Descripción	Valida los datos introducidos y actualiza los datos del área en caso de no existir errores.
Nombre	Buscar áreas de responsabilidad registradas en el sistema.
Descripción	Retorna una lista con las áreas que coincidan con los parámetros de búsqueda introducidos.
Nombre	Eliminar área registrada en el sistema.
Descripción	Valida que no existan otras entidades dependientes del área y lo elimina en caso de no existir errores.

Tabla 11: Descripción de la clase NomTipoMedioInformatico.

Tipo de clase: Entidad	
Nombre: NomTipoMedioInformatico	**Propietario:** Juan Ignacio Morales Pestana
Atributo	**Tipo**

denominacion	String
descripcion	String
idEstado	NomEstado
Para cada rasgo asociado	
Nombre	Adicionar un nuevo tipo de medio tecnológico al sistema.
Descripción	Valida los datos introducidos y registra el nuevo tipo de medio informático en caso de no existir errores.
Nombre	Modificar tipo de medio tecnológico registrado en el sistema.
Descripción	Valida los datos introducidos y actualiza los datos del tipo de medio tecnológico en caso de no existir errores.
Nombre	Buscar tipos de medios tecnológicos registrados en el sistema.
Descripción	Retorna una lista con los tipos de medios tecnológicos que coincidan con los parámetros de búsqueda introducidos.
Nombre	Eliminar tipo de medio tecnológico registrado en el sistema.
Descripción	Valida que no existan otras entidades dependientes del medio tecnológico y lo elimina en caso de no existir errores.

2.4.4 Diagrama Entidad-Relación

A través del modelo de datos se definen los conceptos que se manejan en el sistema y que sirven para describir la estructura de la base de datos diseñada. Es decir, en dicho modelo se representan los datos, sus atributos y tipos, sus relaciones y las restricciones que deben cumplirse sobre ellos.

La normalización de bases de datos es una técnica que puede ayudar a evitar la aparición de anomalías en los datos así como otras cuestiones de administración de los mismos. Esta técnica consiste en transformar una tabla en varias fases: primera forma normal, segunda forma normal, tercera forma normal y otras. El objetivo consiste en[41]:

- Eliminar la redundancia de los datos y en consecuencia utilizar menos espacio.
- Facilitar la tarea de realizar cambios en los datos y evitar las anomalías al hacerlo.

- Facilitar la implementación de los requisitos de integridad referencial.
- Generar una estructura fácilmente comprensible muy parecida a la situación que representan los datos y que permite su crecimiento.

A continuación se presenta el diagrama entidad-relación, el mismo se encuentra en tercera forma normal.

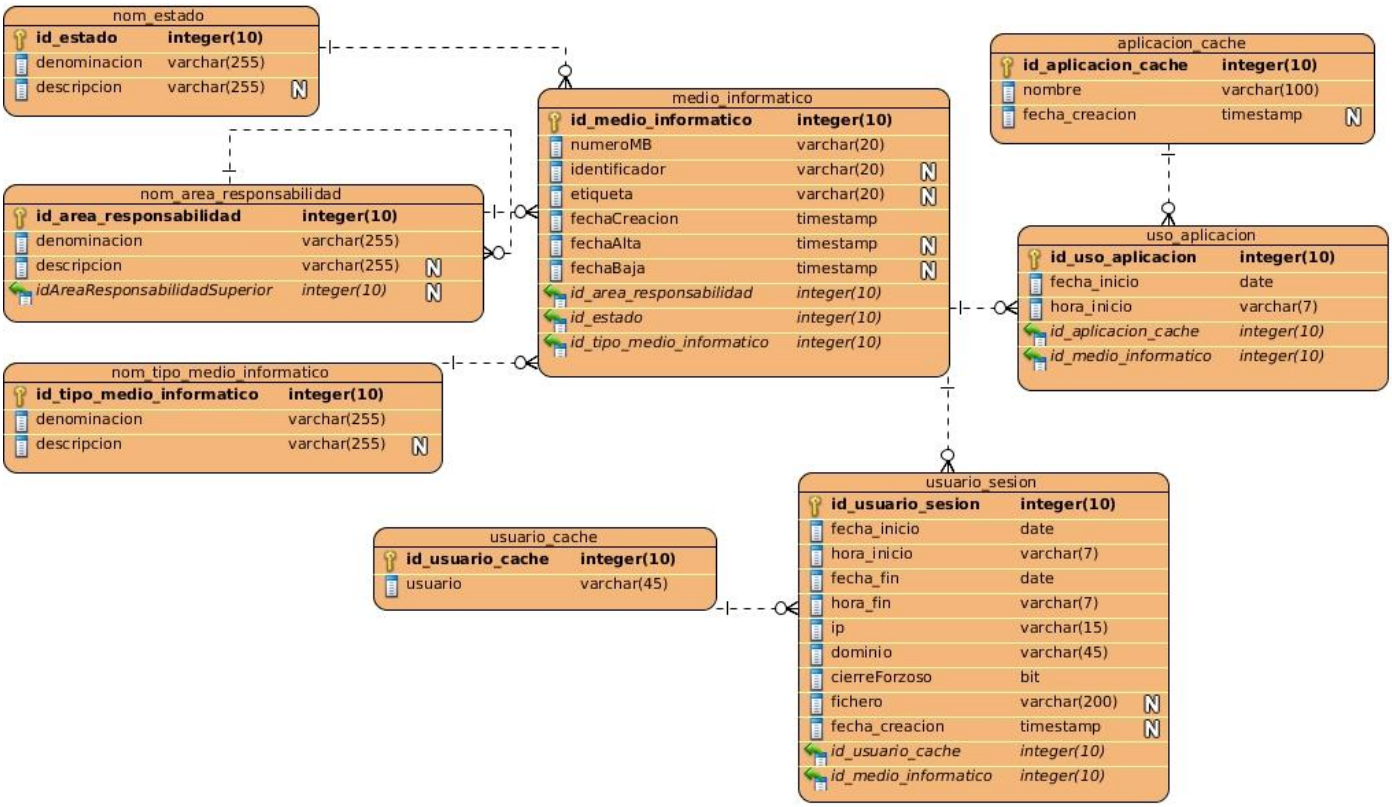

Ilustración 4: Diagrama Entidad-Relación.

Descripción de las tablas

Se identificaron ocho tablas, a continuación se detallan dos de las mismas; el resto se encuentra disponible en el anexo ***DT-CEUS-Descripcion-DER.doc***.

Tabla 12: Descripción de la tabla nom_area_responsabilidad.

Nombre: nom_area_responsabilidad

Descripción: Contiene las diferentes áreas de la organización.		
Atributo	**Tipo**	**Descripción**
denominacion	varchar	Nombre del estado.
descripcion	varchar	Descripción del estado
asset	varchar	Identificador del área según el Departamento de Economía y Contabilidad (DEC).
idAreaResponsabilidadSuperior	integer	Llave foránea que referencia a la tabla nom_area_responsabilidad.

Tabla 13: Descripción de la tabla medio_informatico.

Nombre: medio_informatico		
Descripción: Contiene los medios informáticos.		
Atributo	**Tipo**	**Descripción**
numeroMB	varchar	Identificador del medio según el DEC.
identificador	varchar	Identificador del medio adicional para uso exclusivo del sistema.
etiqueta	varchar	Nombre por el cual se visualizará el medio en caso de poseerlo.
fecha_creacion	timestamp	Fecha en que fue registrado el medio en el sistema.
fecha_alta	date	Fecha en que el medio fue entregado a la organización.
fecha_baja	timestamp	Fecha en que el medio es retirado de la organización.
id_area_responsabilidad	integer	Llave foránea que referencia al identificador de la tabla nom_area_responsabilidad.
id_estado	integer	Llave foránea que referencia al identificador de la tabla nom_estado.
id_tipo_medio_informatico	integer	Llave foránea que referencia al identificador de la tabla nom_tipo_medio_informatico.

2.4.5 Diagramas de secuencia

Un diagrama de interacción explica gráficamente las interacciones existentes entre las instancias (y las clases) del modelo de estas. El UML define dos tipos de estos diagramas; ambos sirven para expresar interacciones semejantes o idénticas de mensajes. Los diagramas de colaboración describen las interacciones entre los objetos en un formato de grafo o red. Los diagramas de secuencia describen las interacciones en una especie de formato de cerca o muro[36].

Para el diseño de la propuesta de solución se elaboraron 32 diagramas de secuencia, a continuación se presenta el diagrama de secuencia del rasgo *Adicionar un medio tecnológico al sistema*.

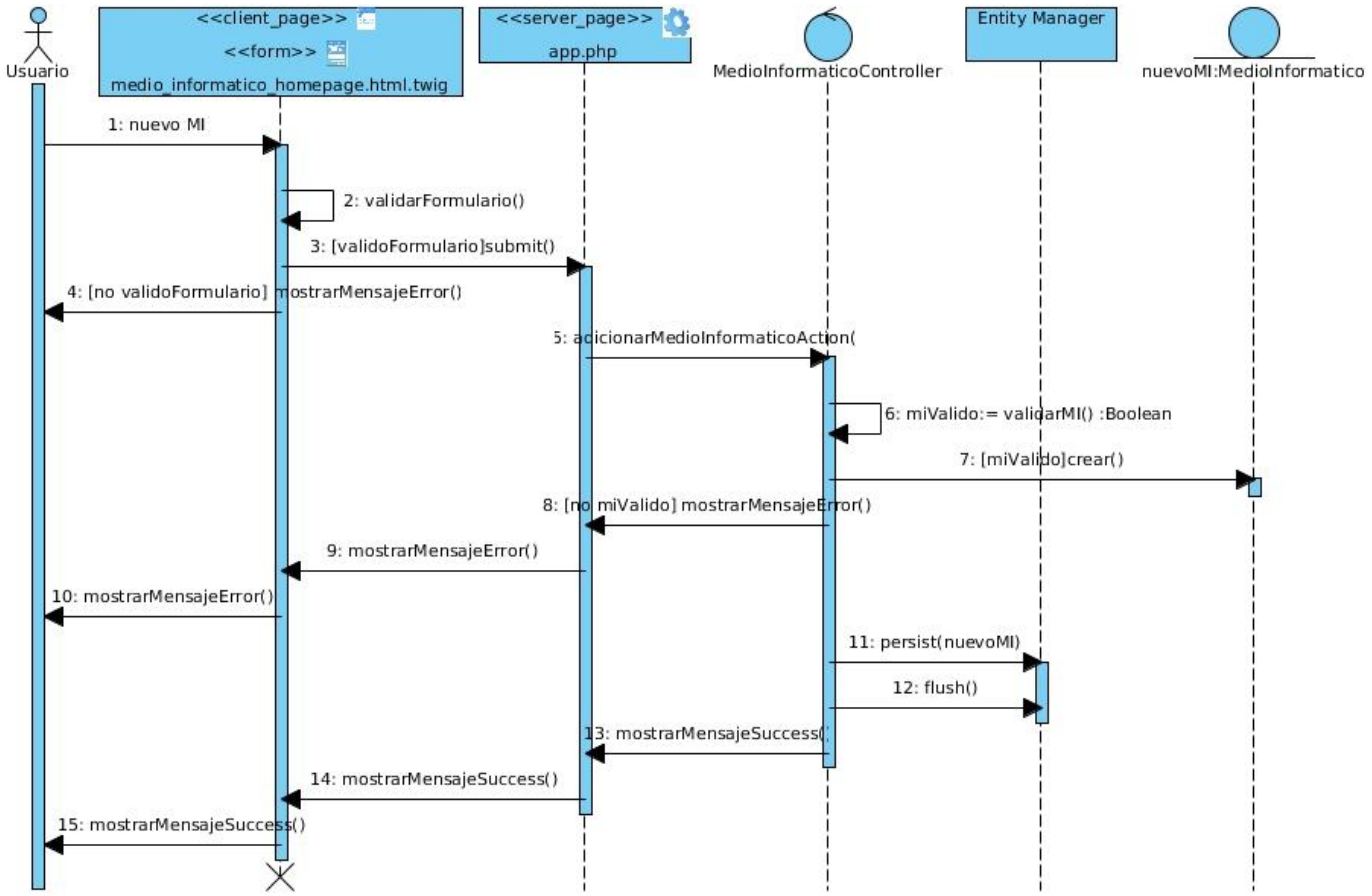

Ilustración 5: Diagrama de secuencia de FS3R9.

2.5 Implementación de la solución

2.5.1 Definición de estándares

Estándares de diseño de base de datos

El uso de estos estándares presenta ventajas como:

- Asegurar la legibilidad del modelo de datos, inclusive para personas que no están relacionadas con el ambiente informático, en etapas de análisis y diseño.
- Facilitar la portabilidad entre motores de bases de datos, plataformas y aplicaciones.
- Facilitar la tarea de los programadores en el desarrollo de los sistemas.

Es por esto que la codificación de las tablas de las bases de datos a desarrollar debe cumplir ciertos requisitos, detallados en el documento adjunto ***Estándares de diseño de bases de datos.pdf***. Estos requisitos pueden aplicarse a cualquier motor de bases de datos.

Estándares de codificación

En la actualidad existen diferentes estándares de codificación cuya utilización favorece la comunicación fluida y directa entre los desarrolladores, permitiendo el aumento de la reutilización y el mantenimiento de los sistemas.

Para implementar la solución propuesta se usan los estándares establecidos, además de los que ya utiliza el marco de trabajo. Dichos estándares se describen a continuación:

Generales

- Todos los nombres serán definidos haciendo uso de la notación CamelCase en cualquiera de sus dos versiones (lower o Upper).
- Los nombres deberán ser definidos con la menor cantidad de palabras posibles, evitando en la medida de lo posible el uso de siglas.

Acciones[27]

- Dentro de las especificaciones del marco de trabajo se encuentra que cada una de las acciones debe terminar con la palabra **Action**.
- Todos los nombres de las acciones deben ser definidos haciendo uso de la nomenclatura **lowerCamelCase**.
- En caso de ser acciones referentes a la eliminación, adición o modificación de un elemento se deben usar nombre específicos como **eliminarAction**.
- Los nombres de las acciones se deben definir con la menor cantidad de palabras posibles y de ser posibles estar en infinitivo.

Ejemplo: adicionarAreaResponsabilidadAction, buscarAreaResponsabilidadAction.

Clases[28]

- Todos los nombres de las clases deben ser definidos haciendo uso de la notación **UpperCamelCase**.
- Debe expresar con claridad cuál es el objetivo y el alcance de la clase.

Ejemplo: AreaResponsabilidadController, MedioInformaticoRepository

Funciones

- Todos los nombres de las funciones serán definidos haciendo uso de la notación **lowerCamelCase**.

Ejemplo: findAreasActivas, findAreaById

Variables

[27]Cualquier función de visibilidad pública contenida en una clase controladora destinada a la interacción directa con la vista.

[28]Clases controladoras, entidades y otras clases.

- Los nombres de las variables deben expresar claramente el contenido de las mismas y serán definidos según la notación **lowerCamelCase**.
- Se definen al principio de la estructura donde serán utilizadas en la medida de lo posible.
- Se excluyen de esta nomenclatura las variables generadas por el propio marco de trabajo.

Ejemplo: nuevaArea, errores.

2.5.2 Tratamiento de errores

Un aspecto importante a tener en cuenta para desarrollar un software es el tratamiento de errores, tanto los generados por los usuarios como los producidos por el propio sistema.

Las validaciones realizadas del lado del servidor se encargan de controlar el flujo de los datos recibidos en el controlador para evitar la inconsistencia de la información almacenada en la base de datos. Las validaciones realizadas en las vistas se encargarán de que el usuario haga una entrada correcta o parcialmente correcta de los datos. Tienen como función principal evitar que se introduzcan datos incorrectos en los diferentes campos de los formularios para impedir que datos inconsistentes o incorrectos lleguen al servidor.

Con el objetivo de detectar y mostrar las alertas y mensajes de notificación de errores en la interfaz del sistema se utilizan las validaciones que provee jQuery haciendo uso de notificaciones. En cada acción implementada en Symfony se realiza una revisión de los valores de las variables que llegan al servidor, haciendo uso de los validadores[29]de las entidades y en caso de detectarse errores son enviados en forma de cadena JSON[30]a la interfaz para que sean notificados al usuario.

Por otra parte, los mensajes de error sólo contendrán la información necesaria para que el usuario comprenda lo que ha ocurrido, nunca se deberán mostrar detalles que puedan comprometer la integridad del sistema. En caso de que las peticiones realizadas vía URL generen una excepción o error el usuario será redirigido hacia una página donde se mostrará la información necesaria para que conozca el error que ha ocurrido; en caso de la petición ser asincrónica será notificado mediante un mensaje.

[29] Grupo de clases que permiten establecer determinadas restricciones sobre las entidades con el fin de hacerlas válidas para el sistema. Haciendo uso de esta herramienta las entidades se podrían validar por si solas.
[30] Java Script Object Notation.

2.5.3 Diagrama de componentes

Este tipo de diagrama muestra el conjunto de componentes y sus relaciones entre sí, representando cómo un sistema de software es dividido, permitiendo además, describir la vista de implementación estática de un sistema. [13]

A continuación se muestra el diagrama de componentes del sistema.

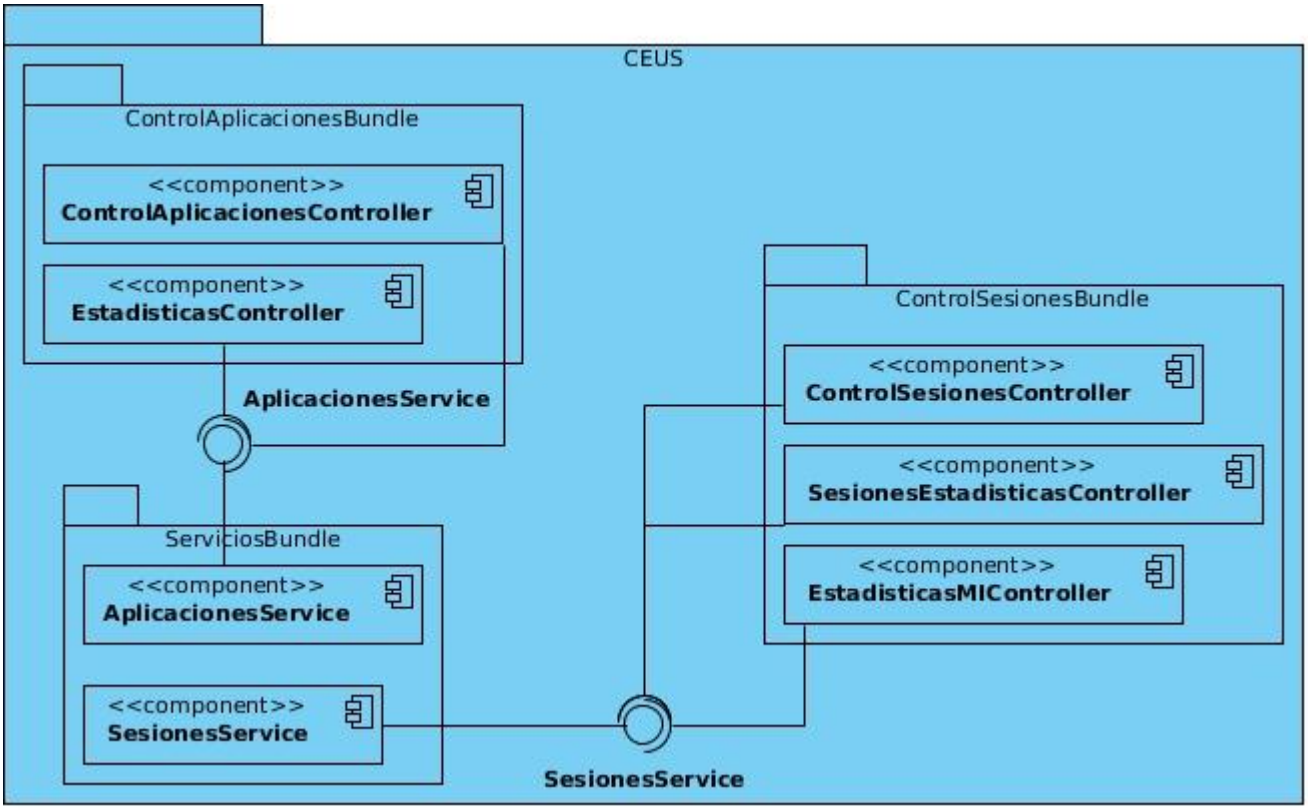

Ilustración 6: Diagrama de componentes de la solución.

Servicios brindados por la clase AplicacionesService:

- getDatosAplicacion: Se obtienen los datos de uso de una aplicación en un área determinada.
- buscarAplicaciones: Se obtiene un arreglo de uso de aplicaciones según los parámetros recibidos.

Servicios brindados por la clase SesionesService:

- getDatosUsuario: Se obtienen los datos del comportamiento de uso de un usuario en un área determinada.

- getDatosPc: Se obtienen los datos generales de uso de un medio tecnológico.
- getComportamientoPc: Se obtienen los datos del comportamiento de uso diario de un medio tecnológico.
- rankingUsuarioPc: Se obtienen los 10 usuarios más frecuentes en un medio tecnológico con los datos de uso correspondiente a cada uno.
- buscarSesiones: Se obtiene un arreglo de las sesiones de usuario según los parámetros recibidos.

2.5.4 Diagrama de despliegue

El diagrama de despliegue muestra la arquitectura del sistema desde el punto de vista del despliegue de cada uno de los elementos necesarios para el funcionamiento del software, incluido el mismo software. Los artefactos representan elementos del mundo. Ejemplos de artefactos son archivos ejecutables, impresoras, esquemas de bases de datos, archivos de configuración, entre otros.

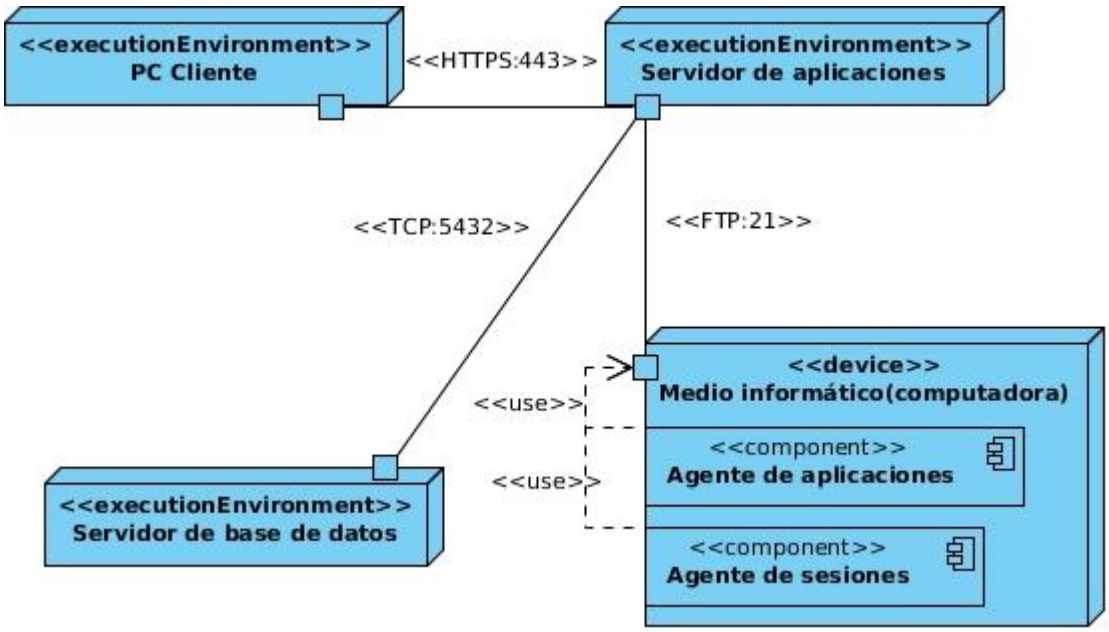

Ilustración 7: Diagrama de despliegue de la solución.

2.6 Conclusiones del capítulo

- La identificación de los rasgos posibilitó el posterior diseño de la solución enfatizando en las características funcionales que este debe cumplir.
- El empleo de las técnicas de captura de requisitos facilitó la elaboración del listado de características funcionales del sistema, permitiendo identificar los aspectos del dominio y los recursos de aplicación y de soporte, que definirán la estructura del sistema que se pretende desarrollar como propuesta de solución.
- El actual capítulo fue decisivo para la definición de la arquitectura de la solución que se expresa y el diseño de los requerimientos funcionales identificados. Se fundamentaron los patrones de diseño empleados que permitieron especificar la estructura, comportamiento y relaciones de las clases, garantizando presentar una solución que atiende los niveles de reutilización y mantenimiento.
- La flexibilidad de Symfony permitió establecer una arquitectura orientada a capas haciendo uso del patrón Modelo-Vista-Controlador lo que posibilitó establecer las bases para la implementación de la solución.

Capítulo 3: VALIDACIÓN DE LA SOLUCIÓN

Probar un programa es la forma más común de comprobar que satisface sus especificaciones y que hace lo que el cliente quiere que haga. El proceso de pruebas del software tiene dos objetivos distintos[38]:

- Demostrar al desarrollador y al cliente que el software satisface sus requerimientos.
- Descubrir defectos en el software, en que el comportamiento de este es incorrecto, no deseado o no cumple su especificación.

En este capítulo se realiza la validación de la solución implementada en el capítulo anterior demostrando así la correcta implementación de las funcionalidades y el cumplimiento de las especificaciones del cliente.

3.1 Niveles de prueba

Existen varias estrategias para realizar las pruebas de software las cuales dependen de las características del mismo, a continuación se describen dichas técnicas[13]:

3.1.1 Pruebas unitarias

Las pruebas unitarias centran sus esfuerzos en pequeñas unidades del software (alguna funcionalidad en específico, componente o módulo). Las pruebas unitarias se centran en la lógica de procesamiento de datos y las estructuras internas dentro de los límites de un componente o porción de código. Este tipo de prueba puede llevarse a cabo en paralelo para múltiples componentes.

Consideraciones:

- Se examinan las estructuras de datos locales para garantizar que los datos almacenados mantengan temporalmente su integridad durante todas las etapas de la ejecución de un algoritmo.
- Todos los caminos independientes a través de la estructura de control son ejercitados, comprobando así que todas las sentencias se ejecutarán al menos una vez.
- Las condiciones de límite son probadas para asegurar que el sistema funciona correctamente en los límites establecidos para la restricción del procesamiento; y por último, se ponen a prueba los caminos de manejo de errores.

3.1.2 Pruebas de aceptación

Las pruebas de aceptación comienzan una vez que se ha ensamblado el software como un todo y ya se han probado y ejercitado todos los componentes de forma individual. La prueba se centra en las acciones visibles para el usuario y en la salida del sistema que este sea capaz de reconocer.

La validación del software se logra mediante la aplicación de una serie de pruebas que garantizan que el mismo cumpla con sus especificaciones.

3.1.3 Pruebas de sistema

En un final el software se incorpora a elementos del sistema, como hardware, personas e información y se realizan una serie de pruebas de validación y de integración. Estas pruebas están fuera del alcance del proceso de software y no son realizadas por los ingenieros de software solamente. La prueba de sistema abarca una serie de pruebas diferentes cuyo propósito es ejercitar profundamente el sistema. Aunque cada una de las pruebas tiene un propósito diferente todas trabajan en conjunto para comprobar que se han integrado correctamente todos los componentes del sistema y que el mismo cumple con todas sus especificaciones.

3.2 Validación

3.2.1 Validación del diseño

Existen varios tipos de métricas que pueden ser utilizadas en la realización de proyectos de software para gestionar, predecir y mejorar la calidad de software. La finalidad del uso de métricas es evaluar los sistemas para conseguir alta calidad[42].

A continuación se describe el proceso de validación del diseño haciendo uso de las métricas, Tamaño operacional de clase y Relaciones entre clases.

No	Bundle	Clase	Procedimientos	Relaciones de uso
1	Administración	AdministracionController	3	0
2	Administración	AreaResponsabilidadController	7	1

3	Administración	MedioInformaticoController	7	1
4	Administración	TipoMedioInformaticoController	7	1
5	Administración	NomTipoMedioInformatico	10	0
6	Administración	NomTipoMedioInformaticoRepository	4	0
7	Administración	MedioInformatico	20	0
8	Administración	MedioInformaticoRepository	9	0
9	Administración	NomAreaResponsabilidad	14	0
10	Administración	NomAreaResponsabilidadRepository	2	0
11	Administración	NomEstado	6	0
12	Administración	NomEstadoRepository	10	0
13	ControlAplicaciones	ControlAplicacionesController	4	0
14	ControlAplicaciones	EstadisticasController	10	0
15	ControlAplicaciones	AplicacionesService	2	0
16	ControlAplicaciones	AplicacionesCommand	2	2
17	ControlAplicaciones	SoftwareUse	11	0
18	ControlAplicaciones	SoftwareCache	5	0
19	ControlSesiones	ControlSesionesController	7	0
20	ControlSesiones	EstadisticasMIController	5	0
21	ControlSesiones	EstadisticasUsuarioController	6	0
22	ControlSesiones	SesionesEstadisticasController	2	0
23	ControlSesiones	SesionesService	6	0
24	ControlSesiones	SesionesCommand	2	2
25	ControlSesiones	SesionLogin	23	0
26	ControlSesiones	UsuarioLogin	3	0
27	Servicios	ReportePdfController	6	0

Métrica tamaño operacional de clase (TOC)

El tamaño operacional de las clases está dado por el número de métodos asignados a una clase. Mediante el cual se calcula el nivel de responsabilidad de los métodos, la complejidad de implementación de los mismos y su reutilización a fin de inspeccionar la efectividad del diseño.

El primer paso para la aplicación de esta métrica es identificar la cantidad de procedimientos con los que cuenta cada una de las clases y el promedio de procedimientos entre las mismas.

Total de clases: 27

Promedio de procedimientos: 7,148148148

Luego de obtenido los valores anteriores se procede a aplicar los siguientes criterios en función de la cantidad de procedimientos de cada clase (cp) para la clasificación en Alta, Baja o Media de los atributos de Responsabilidad, Complejidad de implementación y Reutilización.

Tabla 14: Umbrales asociados a la métrica TOC.

	Categoría	Criterio
Responsabilidad	Baja	cp <= promedio
	Media	2*promedio >cp>= promedio
	Alta	cp > 2*promedio
Complejidad	Baja	cp <= promedio
	Media	2*promedio >cp>= promedio
	Alta	cp > 2*promedio
Reutilización	Baja	cp > 2*promedio
	Media	2*promedio >cp>= promedio
	Alta	cp <= promedio

De forma estadística se pueden analizar los promedios de los procedimientos por criterio de cantidad de procedimientos de la siguiente manera:

Tabla 15: Resultado por criterio de cantidad de procedimientos.

Criterio	Cantidad de clases	Porciento
Entre 1 y 5	11	40,74074074
Entre 6 y 10	12	44,44444444
Entre 11 y 15	2	7,407407407
Entre 16 y 20	1	3,703703704
Entre 21 y 25	1	3,703703704
Total	**27**	**100**

A continuación se muestran los resultados de la evaluación de cada uno de los atributos de calidad para la métrica TOC.

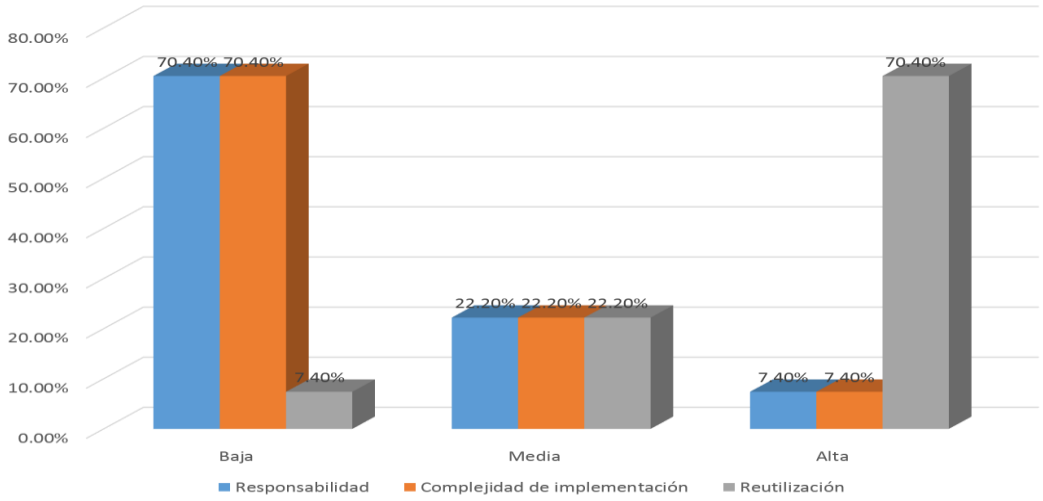

Ilustración 8: Resultado de la evaluación de los atributos para la métrica TOC.

Cuando existe un TOC alto se afectan los parámetros de calidad definidos por esta métrica. Se reduce la reutilización de las clases, aumenta la responsabilidad, la implementación se hace más compleja y las pruebas son difíciles de realizar y aumenta la responsabilidad de las clases.

La mayoría de las clases que conforman el sistema están dentro de las categorías de baja y media, para un 92 % del total, lo que demuestra la elevada reutilización, baja complejidad y responsabilidad en el diseño propuesto, por tanto se concluye que los resultados obtenidos, según esta métrica, son satisfactorios.

Métrica relaciones entre clases (RC)

Esta métrica identifica el número de clases a las cuales una está ligada. Ocurre una dependencia entre dos clases cuando una de ellas usa métodos o variables de la otra.

El primer paso para la aplicación de esta métrica es identificar la cantidad de relaciones de uso con las que cuenta cada una de las clases con respecto a las otras, la cantidad de clases y el promedio de relaciones de uso entre las mismas.

Total de clases: 27

Promedio de relaciones de uso: 0,259259259

Luego de obtenido los valores anteriores se procede a aplicar los siguientes criterios en función de la cantidad de relaciones de uso de cada clase (cru) para la distinta clasificación de los atributos Acoplamiento, Complejidad de mantenimiento, Reutilización y Cantidad de pruebas.

Tabla 16: Umbrales asociados a la métrica RC.

	Categoría	Criterio
Acoplamiento	Ninguno	cru = 0
	Bajo	cru = 1
	Medio	cru = 2
	Alto	cru > 2
Complejidad de mantenimiento	Baja	cru <= promedio
	Media	2*promedio >cru>= promedio
	Alta	cru > 2*promedio
Reutilización	Baja	cru > 2*promedio
	Media	2*promedio >cru>= promedio
	Alta	cru <= promedio
Cantidad de pruebas	Baja	cru <= promedio
	Media	2*promedio >cru>= promedio
	Alta	cru > 2*promedio

De la misma forma se hace un análisis independiente para cada uno de los atributos permitiendo observar con claridad los resultados.

Analizando la categoría de las clases según la cantidad de dependencias.

Tabla 17: Criterio basado en la cantidad de dependencias.

Criterio	Categoría	Cantidad de clases	Porciento
0 dependencias	Muy Bueno	22	81,48148148
1 dependencias	Bueno	3	11,11111111
2 dependencias	Regular	2	7,407407407
3 dependencias	Malo	0	0
> 3 dependencias	Muy Malo	0	0
Total	**-**	**27**	**100**

Tabla 18: Resumen del atributo Reutilización.

Reutilización	Cantidad de clases	Promedio
Baja	5	18,51851852
Media	0	0
Alta	22	81,48148148
Total	**27**	**100**

A continuación se muestran los resultados de la evaluación de cada uno de los atributos de calidad para la métrica RC.

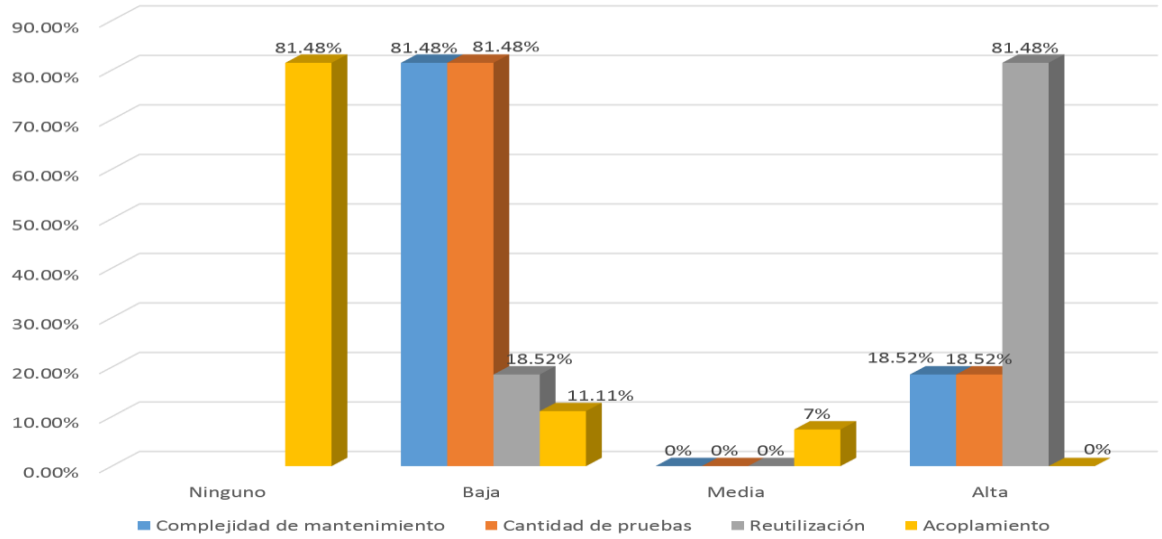

Ilustración 9: Resultado de la evaluación de los atributos para la métrica RC.

Los resultados obtenidos durante la evaluación de esta métrica demuestran que el diseño propuesto se encuentra dentro de los niveles de calidad requeridos. Los atributos de calidad fueron evaluados satisfactoriamente confirmando la elevada reutilización, el bajo acoplamiento, la baja complejidad de mantenimiento y la baja cantidad de pruebas que se necesitan realizar en el diseño propuesto.

3.2.3 Ejecución de las pruebas de caja blanca

Pruebas unitarias

El marco de trabajo Symfony no sólo facilita el trabajo del desarrollador al proveerlo de una arquitectura base y procesos de generación de código automatizados sino que al integrarse con otras herramientas como PHPUnit facilita el proceso de las pruebas unitarias a la aplicación que se construye, además cuenta con poderosas herramientas que permiten no sólo validar las clases por sí mismas, sino también simular peticiones para observar el correcto comportamiento de la aplicación.

Para la realización de las pruebas unitarias se ha seleccionado la versión 4.1.0 de la herramienta.

Validación de las entidades

La validación de los datos de entrada es de suma importancia para que a los controladores lleguen estos datos con la menor cantidad de errores posibles. En ocasiones, no es posible o se dificulta, la validación de ciertos campos en la vista; por lo que llegan al servidor datos parcialmente válidos y es necesario el uso de condicionales, largas expresiones regulares en el código, lo que entorpece su entendimiento. Haciendo uso de los validadores de Symfony se logra que la entidad se valide a sí misma y se reduce el código para hacer esta operación.

```
/**
 * @var string
 *
 * @ORM\Column(name="denominacion", type="string", length=45, nullable=false)
 * @Assert\NotNull(
 *     message = "El campo denominación es requerido."
 * )
 * @Assert\NotBlank(
 *     message = "El campo denominación no puede contener solo espacios."
 * )
 * @Assert\Regex(pattern="/[!@#?^¿_$%\';.,&*(()]/", match=false, message="El campo denominación contiene caracteres no válidos.")
 * @Assert\Length(
 *      min = "5",
 *      max = "15",
 *      minMessage = "El campo denominación debe contener al menos {{ limit }} caracteres.",
 *      maxMessage = "El campo denominación no debe contener más de {{ limit }} caracteres."
 * )
 */
private $denominacion;
```

Ilustración 10: Validadores del campo denominación de la entidad NomAreaResponsabilidad.

La aplicación de esta prueba arrojó como resultado que no se validaba correctamente el campo denominación para las entidades NomAreaResponsabilidad y NomTipoMedioInformatico, y el campo numeroMedioBasico de la entidad MedioInformatico. Las mismas permitían la inserción de valores que

contenían sólo espacios. El error fue corregido haciendo uso de la función trim[31] de PHP para obtener los siguientes resultados:

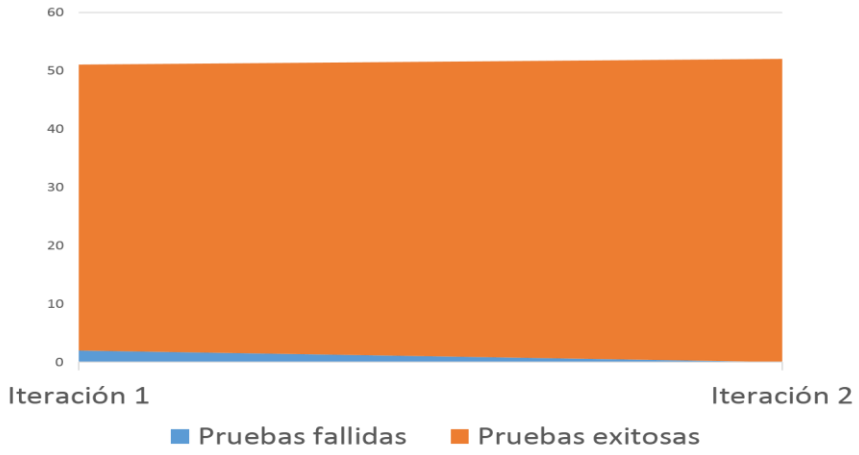

Ilustración 11: Resultado de la aplicación de las pruebas unitarias.

Para más información acerca de la programación de las pruebas unitarias y el resultado de las mismas referirse al anexo ***pruebas_unitarias.tar.gz***.

De esta forma se demuestra la correcta implementación de las clases y de los validadores de las mismas, no permitiendo la entrada al sistema por parte del usuario de clases no válidas.

Prueba de la ruta básica

El método de la ruta básica permite que el diseñador de casos de prueba obtenga una medida de complejidad lógica procedimental y que use esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de pruebas derivados para ejercitar el conjunto básico de rutas debe garantizar que se ejecute cada una de las instrucciones del programa al menos una vez[13].

Esta prueba permite identificar no sólo las instrucciones que no se ejecutan en el código sino que también permite optimizar el mismo al identificar ciclos y condicionales innecesarias; reduciendo así la complejidad ciclomática del mismo y su compresión para posteriores mejoras y mantenimientos.

[31] Función que elimina un carácter del principio y fin de una cadena de texto. Por defecto elimina los espacios.

Para la aplicación de esta prueba es necesario conocer el conjunto de caminos independientes de un algoritmo determinado, para lo cual se construye el grafo de flujo asociado al algoritmo y se calcula su complejidad ciclomática. La complejidad ciclomática se basa en la teoría de grafos y se calcula de una de las tres siguientes maneras[13]:

1. El número de regiones corresponde a la complejidad ciclomática.
2. La complejidad ciclomática, V(G), de una gráfica de flujo, G, se define como:
   2.1 V (G) = E-N+2.
   2.2 Donde *E* es el número de aristas y *N* el número de nodos.
3. La complejidad ciclomática, V(G), de una gráfica de flujo, G, se define como:
   3.1 V (G) = P+1.
   3.2 Donde P es el número de nodos predicados [32]contenidos en la gráfica.

A continuación se presenta la prueba de la ruta básica realizada al método ***adicionarAreaResponsabilidadAction*** de la clase ***AreaResponsabilidadAction***.

[32] Nodos de los cuales se toman diferentes caminos. Condicionales o ciclos.

AreaResponsabilidadController.php

```php
public function adicionarAreaResponsabilidadAction()
    {
        $request = $this->get('request');
        $em = $this->getDoctrine()->getManager();

        $estado = $em->getRepository('AdministracionBundle:NomEstado')->findEstadoActivo();

        $denominacion = $request->get('denominacion');
        $descripcion = $request->get('descripcion');
        $idAreaResponsabilidadSuperior = $request->get('idAreaResponsabilidadSuperior');
        $asset = $request->get('asset');

        $area = new NomAreaResponsabilidad(); //1

        if ($idAreaResponsabilidadSuperior != '') { //2
            $areaSuperior = $em->getRepository('AdministracionBundle:NomAreaResponsabilidad')->findAreaById(
                $idAreaResponsabilidadSuperior
            );
            //3
            if (!$areaSuperior) { //4
                $response = array(
                    'status' => 'error',
                    'msg' => 'El área a la que hace referencia no existe.'
                );

                return new Response(json_encode($response)); //5
            }

            $area->setIdAreaResponsabilidadSuperior($areaSuperior); //6
        }

        $area->setDenominacion(trim($denominacion));
        $area->setDescripcion(trim($descripcion));
        $area->setIdEstado($estado);
        $area->setAsset(trim($asset));
        $area->setFechaCreacion(new \DateTime());

        $errores = $this->get('validator')->validate($area); //7

        if (count($errores) != 0) //8
        {
            $response = array(
                'status' => 'error',
                'msg' => $errores[0]->getMessage()
            );

            return new Response(json_encode($response)); //9
        }

        $em->persist($area);
        $em->flush();

        $response = array(
            'status' => 'success',
            'msg' => 'Área adicionada correctamente.'
        );

        return new Response(json_encode($response)); //10
    }//11
```

Ilustración 12: Instrucciones del método adicionarAreaResponsabilidadAction.

El primer paso es identificar el grafo de flujo asociado al método objeto de la prueba.

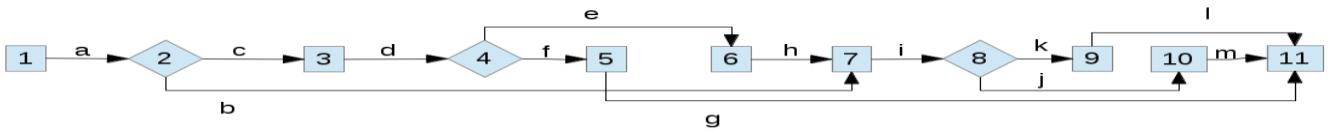

Ilustración 13: Grafo de flujo asociado al método adicionarAreaResponsabilidadAction.

Según las distintas vías para el cálculo de la complejidad ciclomática:

1. La gráfica contiene 4 regiones.
2. V(G) = 13 aristas – 11 nodos + 2 = 4
3. V(G) = 3 nodos predicados + 1 = 4

De los resultados obtenidos se deduce que la complejidad ciclomática es igual a cinco, por lo que el método objeto de prueba contiene cinco caminos básicos. Los caminos identificados son:

1. 1-2-7-8-10-11
2. 1-2-7-8-9-11
3. 1-2-3-4-5-11
4. 1-2-3-4-6-7-8-9-11

Es necesario relacionar los datos de manera tal que se establezcan apropiadamente las condiciones de los nodos predicados a medida que se prueba cada una de las rutas. Cada caso de prueba se ejecuta y se compara con los resultados esperados. Una vez completados los casos se puede asegurar que cada una de las instrucciones del programa se han ejecutado al menos una vez[13].

A continuación se representa la matriz de la gráfica, la cual ayuda a comprender con mayor facilidad como están conectados los nodos identificados, y en consecuencia, en la elaboración de los respectivos casos de prueba para cada uno de los diferentes caminos.

Tabla 19: Matriz del grafo asociado al método adicionarAreaResponsabilidadAction.

Nodo/Conectado a	1	2	3	4	5	6	7	8	9	10	11
1		a									
2			c				b				
3				d							
4					f	e					
5											g
6							h				
7								i			
8									k	j	
9											l
10											m
11											

Con la confección de la matriz del grafo se comprueba que todas las rutas identificadas pueden ser accedidas por lo que se procede a realizar los casos de pruebas correspondientes para cada una.

Tabla 20: Caso de prueba para la ruta básica 1.

Caso de prueba para la ruta básica 1(1-2-7-8-10-11)	
Descripción	Debe crearse un área de responsabilidad sin área de responsabilidad superior.
Condiciones de ejecución	• Todos los datos se introducen correctamente. • No se selecciona un área de responsabilidad superior.
Entrada	Denominación = Prueba 105, Asset = 30105
Resultado esperado	Mensaje informando el éxito de la operación.
Resultado obtenido	Satisfactorio

Tabla 21: Caso de prueba para la ruta básica 2.

Caso de prueba para la ruta básica 2(1-2-7-8-9-11)	
Descripción	El sistema informa acerca de un error cuando valida los datos.
Condiciones de ejecución	• Se introducen valores erróneos que no cumplan los límites de longitud de caracteres establecidos. • No se selecciona un área de responsabilidad superior.
Entrada	Denominación = A, Asset = 30222, Descripción = ddd

Resultado esperado	Mensaje de error del sistema informando acerca del error ocurrido.
Resultado obtenido	Satisfactorio

Para la realización de este caso de prueba se debe eliminar el área Prueba 105.

Tabla 22: Caso de prueba para la ruta básica 3.

Caso de prueba para la ruta básica 3(1-2-3-4-5-11)	
Descripción	El sistema informa acerca de que el área de responsabilidad superior seleccionada no existe.
Condiciones de ejecución	• Todos los datos se introducen correctamente. • Se selecciona un área de responsabilidad superior. • El área de responsabilidad superior a la que se hace referencia debe haber sido eliminada en el momento en que el usuario está realizando la acción.
Entrada	Denominación = Área error, Asset = 30105error, Área superior = Prueba 105.
Resultado esperado	Mensaje del sistema informando acerca de la no existencia del área a la que se hace referencia.
Resultado obtenido	Satisfactorio

Tabla 23: Caso de prueba para la ruta básica 4.

Caso de prueba para la ruta básica 4(1-2-3-4-6-7-8-9-11)	
Descripción	El sistema informa acerca de un error cuando valida los datos.
Condiciones de ejecución	• Todos los datos se introducen correctamente. • Se selecciona un área de responsabilidad superior. • El campo denominación o asset deben ser el de otra Área de responsabilidad.
Entrada	Denominación = Pru, Asset = 30105, Área superior = Laboratorio 101
Resultado esperado	Mensaje de error del sistema informando acerca del error ocurrido.
Resultado obtenido	Satisfactorio

Con la aplicación de esta prueba se pudo demostrar la correcta implementación de la muestra de funcionalidades seleccionadas. Además se identificó otro grupo de funcionalidades que a pesar de ser su funcionamiento el esperado, las mismas no estaban implementadas de forma óptima.

Luego de identificar dichas funcionalidades las mismas se modificaron dando como resultado una reducción media de diez nodos, lo que disminuyó por consiguiente la complejidad de futuro mantenimiento y aumentó la compresión de las mismas.

3.2.4 Ejecución de las pruebas de caja negra

Pruebas de desempeño

Las pruebas de desempeño se aplican para descubrir problemas de desempeño que se representan debido a la falta de recursos del lado del servidor, capacidades inadecuadas de bases de datos y funcionalidades mal diseñadas que pueden conducir a un pobre desempeño cliente-servidor[13].

La solución implementada procesa gran cantidad de información, aumentando a diario el volumen de la misma. Con cada una de las búsquedas realizadas y las gráficas mostradas se le retornan al usuario gran parte o toda la información [33]existente en la base de datos. Debido a esta situación se deben comprobar los tiempos de respuesta de la aplicación ante la solicitud de grandes cantidades de información por parte del usuario. El objetivo de esta prueba no es simular un gran número de peticiones sino simular peticiones que requieran la extracción de gran cantidad de datos.

Para la realización de una prueba de esta índole fue necesaria la investigación de un grupo de herramientas que son capaces de automatizar y apoyar el proceso, las mismas se presentan a continuación[43]:

QALoad

QALoad de la empresa Compuware es una herramienta de automatización de pruebas de carga para aplicaciones web, Java, NET, aplicaciones ERP (Planificación de recursos empresariales) y CRM (gestión de relaciones con los clientes y ambientes distribuidos). Simula miles de usuarios desempeñando

[33] En el caso de las gráficas de comportamiento de las áreas, usuarios y medios tecnológicos se devuelve toda la información existente.

transacciones de negocio clave de una aplicación para asegurar su desempeño y escalabilidad previa a su puesta en producción. Con esta herramienta, los equipos de prueba pueden rápidamente detectar problemas, optimizar el desempeño de los sistemas y ayudar a asegurar un despliegue de aplicaciones exitoso.

JMeter

Es una herramienta Java desarrollada dentro del proyecto Jakarta, que permite realizar Pruebas de Rendimiento y Pruebas Funcionales sobre Aplicaciones Web, permite la ejecución de pruebas distribuidas entre distintos ordenadores, para realizar pruebas de rendimiento. Además activar o desactivar una parte del test, lo que es muy útil cuando se está desarrollando un test largo y se desea deshabilitar ciertas partes que sean muy pesadas o largas. Tiene la forma de generar un caso de prueba a través de una navegación de usuario.

De las herramientas presentadas se seleccionó **JMeter** ya que es una herramienta libre, gratis, dinámica, permite el manejo de los datos de entrada para la realización de las pruebas, guardar los resultados de las mismas en formato de tablas y gráficas para su posterior estudio más detallado. La versión seleccionada de la herramienta es la 2.7.1.

Selección de la muestra de funcionalidades

Para la realización de esta prueba se seleccionaron cuatro rasgos críticos:

1. Buscar sesiones de usuarios registradas en el sistema.
2. Buscar uso de aplicaciones registradas en el sistema.
3. Mostrar estadísticas de aplicaciones más usadas.
4. Mostrar estadística resumen de uso de las áreas de responsabilidad.

Las características de la terminal de trabajo donde se realizarán las pruebas son las siguientes:

1. **Memoria RAM:** 2GB DDR3
2. **Microprocesador:** Celeron 2.06 GHz
3. **Disco duro:** 1TB
4. **Sistema operativo:** Debian Wheezy
5. **Servidor de aplicaciones:** Apache 2.2

6. **Gestor de base de datos:** PostreSQL 9.1
7. **Versión de JRE[34] para JMeter:** 7

Resultado de las pruebas

Las primeras pruebas se realizaron con una cantidad de diez peticiones arrojando resultados satisfactorios. A continuación se muestra la tabla que soporta dicho resultado.

Tabla 24: Resultado de prueba de desempeño con 10 peticiones.

	Muestras	Media	Mínimo	Máximo
Estadísticas aplicaciones	10	283	88	458
Estadísticas generales	10	602	164	771
Aplicaciones búsqueda	10	431	107	563
Sesiones búsqueda	10	359	120	612
TOTAL	**40**	**419**	**88**	**771**

De la misma forma se procedió a realizar las pruebas con 20 peticiones, arrojando los siguientes resultados:

Tabla 25: Resultado de prueba de desempeño con 20 peticiones.

	Muestras	Media	Mínimo	Máximo
Estadísticas aplicaciones	20	1242	100	2691
Estadísticas generales	20	944	687	1251
Aplicaciones búsqueda	20	650	429	972
Sesiones búsqueda	20	680	338	1011
TOTAL	**80**	**879**	**100**	**2691**

[34] Del inglés Java Runtime Environment.

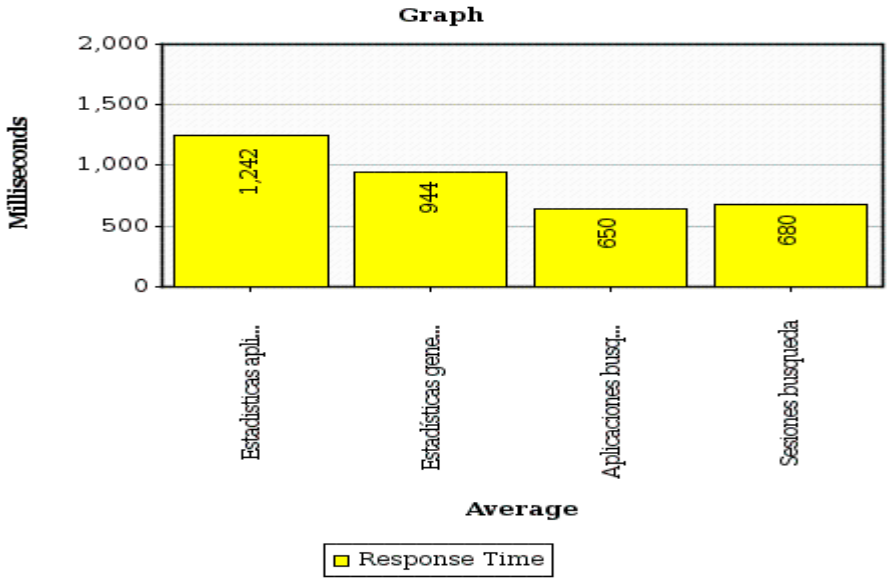

Ilustración 14: Tiempo de respuesta promedio para 20 peticiones.

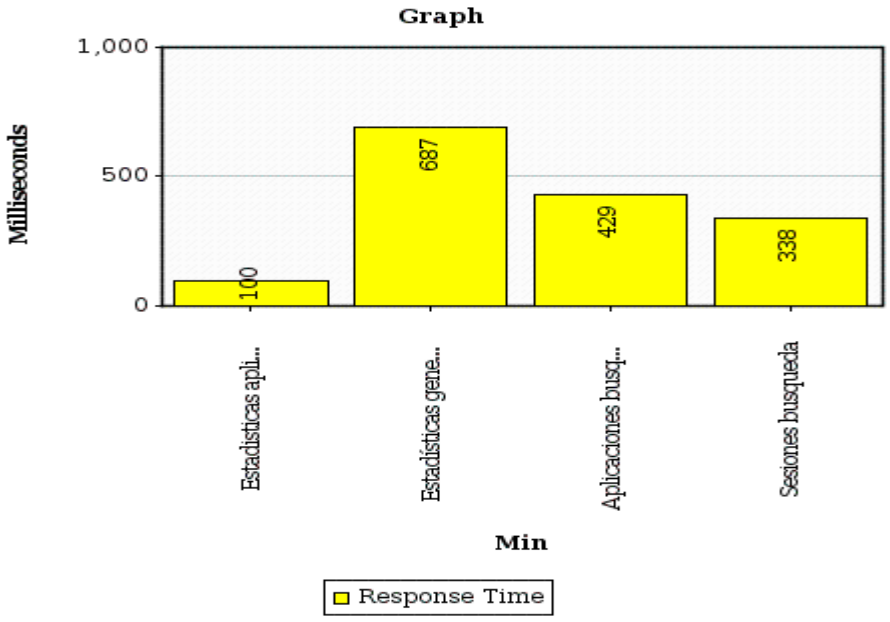

Ilustración 15: Tiempo de respuesta mínimo para 20 peticiones.

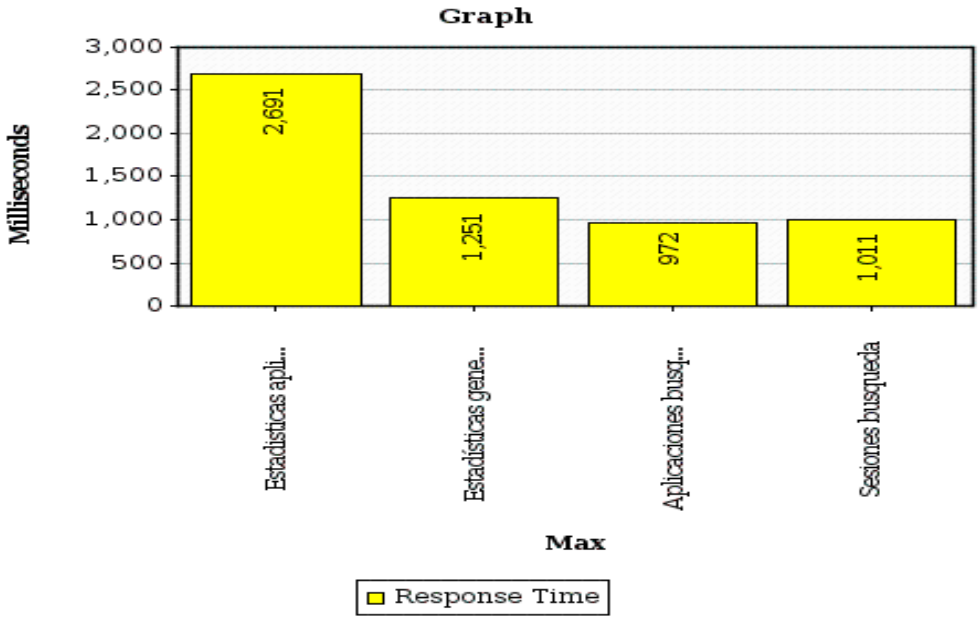

Ilustración 16: Tiempo de respuesta máximo para 20 peticiones.

Como se puede observar, los resultados obtenidos son satisfactorios ya que se sometió al sistema al procesamiento de grandes volúmenes de datos realizando peticiones concurrentes y el mismo mantuvo la media de respuesta por debajo de los dos segundos. La información acerca de la configuración de esta prueba se encuentra disponible en el anexo ***DT-CEUS-Jmeter.jmx***.

Pruebas de aceptación

Para realizar la validación de las pruebas de aceptación se elaboraron un total de 30 pruebas, de las cuales 24 resultaron satisfactorias, dos no satisfactorias y cuatro no realizadas. Los errores detectados y las recomendaciones hechas por el cliente fueron errores de validación y formas incorrectas o confusas de mostrar la información. Las funcionalidades señaladas fueron revisadas y corregidas para una segunda iteración. Los resultados fueron satisfactorios. A continuación se muestra la gráfica que soporta los resultados.

Para más información acerca de los diseños de las pruebas de aceptación se puede consultar el anexo ***DT-CEUS-Pruebas de aceptación.tar.gz***.

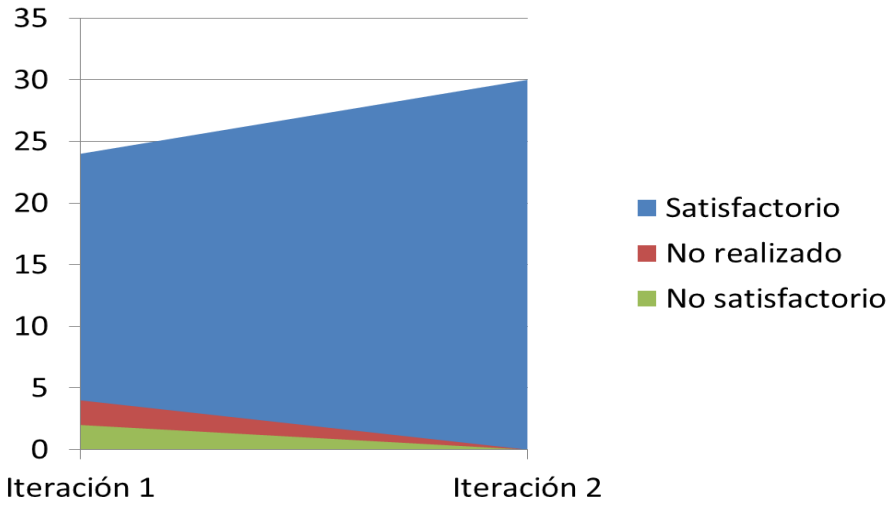

Ilustración 17: Resultado de la aplicación de las pruebas de aceptación.

3.2.5 Validación de la hipótesis

Para la validación de la hipótesis, se seleccionaron 25 planillas de registro de acceso a los laboratorios encontrando un total de 1454 registros. Para lograr controlar el uso de los medios tecnológicos, según la definición dada en la presente investigación, se procedió a identificar los siguientes elementos:

- Usuarios que hacen uso de los medios tecnológicos.
- Horarios de entrada y salida.
- Aplicaciones ejecutadas en los medios tecnológicos.

De igual forma se realizó una búsqueda en el sistema acotando los rangos de fechas a los encontrados en las planillas.

La siguiente gráfica muestra el contraste de los datos capturados en el proceso de control de uso antes y después de la implementación del sistema.

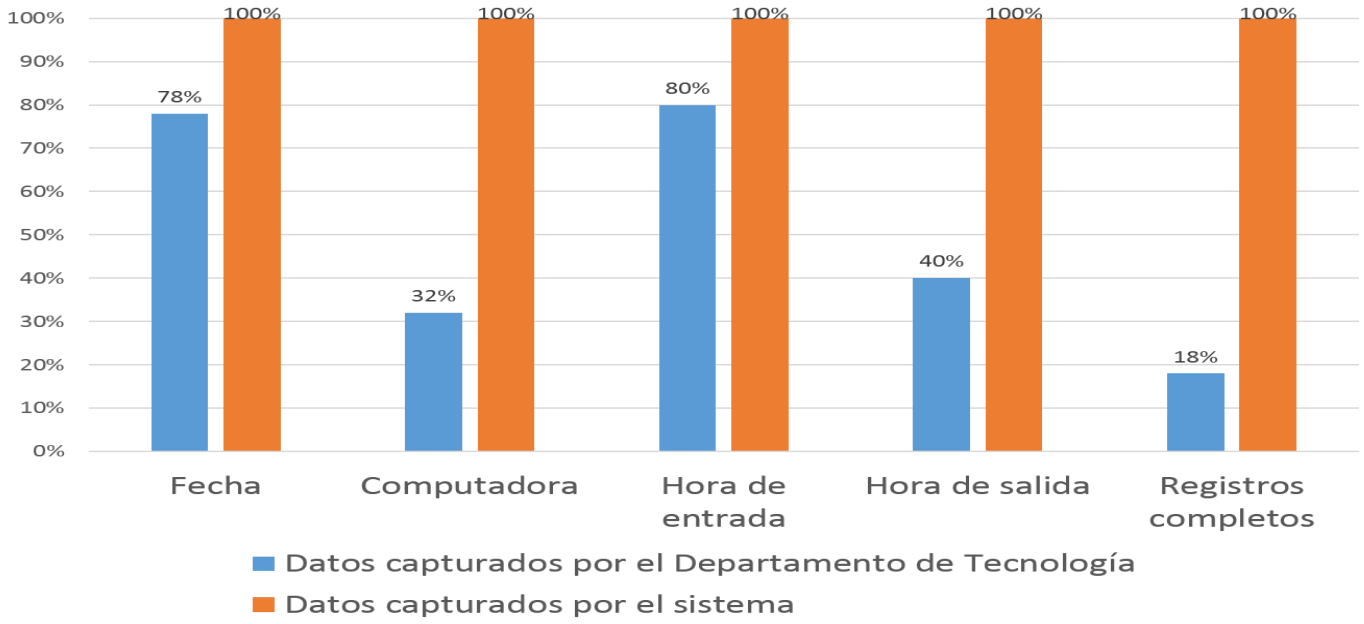

Ilustración 18: Control de uso antes y después del sistema.

Como se muestra en la gráfica, existe una gran cantidad de datos faltantes en cuanto al campo Computadora impidiendo la identificación del medio que es utilizado por los usuarios. Otro de los campos que se manifiesta de esta forma es el campo que registra la hora de salida. De manera general, mediante la muestra de planillas seleccionadas no se puede realizar el control de uso según lo especificado para la presente investigación ya que en más del 50% de las ocasiones no se pueden determinar los elementos a tener en cuenta para el mismo. Otro punto que se debe destacar es que sólo 256 registros de los identificados estaban completos.

Mediante la comparación de los datos recopilados de ambas fuentes se evidencia que el sistema desarrollado es más preciso en el control de uso de los medios tecnológicos ya que se puede determinar qué usuario utilizó qué computadora, en qué momento comenzó a usarla y cuándo dejó de hacerlo; de la misma forma se pueden conocer las aplicaciones ejecutadas en dicha computadora.

3.3 Conclusiones del capítulo

- El uso de las métricas Tamaño Operacional de la Clase y Relaciones entre Clases, permitieron evaluar especialmente la complejidad en la implementación de la solución en la actual investigación.
- Se realizaron pruebas de caja negra y caja blanca para comprobar el correcto funcionamiento de la solución propuesta. Luego de dos iteraciones de realización de pruebas se detectaron un total de cinco no conformidades, las cuales fueron resueltas permitiendo cumplir con los requerimientos identificados.
- Las pruebas de aceptación y la validación han permitido verificar y validar los requerimientos identificados. Estas pruebas demuestran que la solución propuesta soluciona el problema de control de uso de los laboratorios docentes de la Facultad 3.

CONCLUSIONES

El desarrollo de la reciente investigación junto con los resultados obtenidos por la misma, han permitido llegar a las siguientes conclusiones:

- El uso de indicadores esclareció el análisis de soluciones que implementan dentro de sus funcionalidades algunos de los elementos para el control de uso.
- La definición de la solución facilitó su futura implantación para la recuperación de la información que se genera durante el control de uso de los medios tecnológicos.
- La solución desarrollada permite informatizar el proceso de control de uso que se realizaba utilizando planillas de registros, las cuales presentaban riesgos de pérdida y deterioro material dificultando así el registro y consulta de la información.

RECOMENDACIONES

Se recomienda extender las funcionalidades del sistema para alcanzar los siguientes objetivos:

- Monitoreo en tiempo real del control de uso de los medios tecnológicos.
- Sincronización con horarios docentes para medir el aprovechamiento de las clases de laboratorios en función de las aplicaciones utilizadas.

REFERENCIAS BIBLIOGRÁFICAS

[1] G. Restrepo González, “El Concepto y Alcance de la Gestión Tecnológica,” 22-May-2014. [Online]. Available: http://ingenieria.udea.edu.co/producciones/guillermo_r/concepto.html. [Accessed: 22-May-2014].

[2] “Real Academia Española. Diccionario Usual. Definición de recurso.,” 23-May-2014. [Online]. Available: http://lema.rae.es/drae/srv/search?id=iArz5kNDCDXX2vwvZKGg. [Accessed: 23-May-2014].

[3] “Real Academia Española. Diccionario Usual. Definición de tecnología,” 23-May-2014. [Online]. Available: http://lema.rae.es/drae/srv/search?id=ySoh55WGaDXX2xvVnSqu. [Accessed: 23-May-2014].

[4] “Real Academia Española. Diccionario Usual. Definición de medio.,” 23-May-2014. [Online]. Available: http://lema.rae.es/drae/srv/search?id=WYQBczua1DXX2FWZqDnj. [Accessed: 23-May-2014].

[5] “Real Academia Española. Diccionario Usual. Definición de control.,” 23-May-2014. [Online]. Available: http://buscon.rae.es/drae/srv/search?val=control. [Accessed: 23-May-2014].

[6] “Diccionario de la lengua española | Real Academia Española. Definición de uso,” 23-May-2014. [Online]. Available: http://lema.rae.es/drae/?val=uso. [Accessed: 23-May-2014].

[7] “OCS Inventory NG | Home,” 23-May-2014. [Online]. Available: http://www.ocsinventory-ng.org/en/. [Accessed: 23-May-2014].

[8] “Network Inventory Advisor,” 09-Feb-2014. [Online]. Available: http://www.network-inventory-advisor.com/. [Accessed: 09-Feb-2014].

[9] “Aranda SOFTWARE Aranda SOFTWARE,” 22-May-2014. [Online]. Available: http://arandasoft.com/. [Accessed: 22-May-2014].

[10] Y. Ordoñes L., E. Avilés V., J. Hernández P., and O. Rodríguez H., “GRHS. Gestor de Recursos de Hardware y Software.,” Universidad de las Ciencias Informáticas, Centro de Telemática, La Habana, Cuba, 2014.

[11] CISED, “Control de Acceso Entorno Seguro de Desarrollo TICs,” Universidad de las Ciencias Informáticas, Centro de Identificación y Seguridad Digital, La Habana, Cuba.

[12] “Facultad 3. Propuesta de informatización.” 15-Jun-2011.

[13] R. S. Pressman, *Software Engineering. A practitioner´s approach.* 7ma ed. 2010.

[14] J. Rumbaugh, I. Jacobson, and G. Booch, *El lenguaje unificado de modelado. Manual de referencia.* 1998.

[15] I. Jacobson, *Applying UML in The Unified Process*. 1998.

[16] P. Letelier and M. del C. Penadés, “Metodologías Ágiles en el desarrollo de software,” DSIC-Universidad Politécnica de Valencia.

[17] S. Khramtchenko, “Comparing eXtreme Programming and Feature Driven Development in academic and regulated environments,” Harvard University, Final paper for CSCIE-275: Software Architecture and Engineering, May 2004.

[18] A. Molpeceres, “Procesos de desarrollo: RUP, XP y FDD,” Dec. 2002.

[19] “Unified Modeling Language (UML),” 23-May-2014. [Online]. Available: http://www.uml.org/. [Accessed: 23-May-2014].

[20] “PHP: Hypertext Preprocessor,” 09-Dec-2013. [Online]. Available: http://www.php.net/. [Accessed: 09-Dec-2013].

[21] “What is Bourne Again Shell (Bash)? - Definition from Techopedia,” 22-May-2014. [Online]. Available: http://www.techopedia.com/definition/3520/bourne-again-shell-bash. [Accessed: 22-May-2014].

[22] C. Albing, C. Newbam, and J. Vossen, *bash Cookbook*, 1st ed. 2007.

[23] J. Eguiluz, *Desarrollo web ágil con Symfony2*. 2013.

[24] Eduardo Lázaro Valdés Izaguirre, “Componente para el consumo de servicios telemáticos en aplicaciones desarrolladas con symfony 2 en la UCI,” Universidad de las Ciencias Informáticas.

[25] “The technological benefits of Symfony in 6 easy lessons - Symfony,” 11-Feb-2014. [Online]. Available: http://symfony.com/six-technical-reasons. [Accessed: 11-Feb-2014].

[26] “What is Object-Relational Mapping (ORM)? - Definition from Techopedia,” 22-May-2014. [Online]. Available: http://www.techopedia.com/definition/24200/object-relational-mapping--orm. [Accessed: 22-May-2014].

[27] Doctrine Project Team, *Doctrine2 ORM Documentation*. 2011.

[28] “Welcome to the Doctrine Project — Doctrine-Project,” 09-Dec-2013. [Online]. Available: http://www.doctrine-project.org/. [Accessed: 09-Dec-2013].

[29] “About · Bootstrap,” 09-Dec-2013. [Online]. Available: http://getbootstrap.com/about/. [Accessed: 09-Dec-2013].

[30] “jQuery Foundation,” 09-Dec-2013. [Online]. Available: http://jquery.org/. [Accessed: 09-Dec-2013].

[31] “Visual Paradigm - Software Design Tools for Agile Teams.” [Online]. Available: http://www.visual-paradigm.com/. [Accessed: 30-Apr-2014].

[32] “Welcome to NetBeans,” 22-May-2014. [Online]. Available: https://netbeans.org/. [Accessed: 22-May-2014].

[33] “NetBeans IDE - Overview,” 23-May-2014. [Online]. Available: https://netbeans.org/features/index.html. [Accessed: 23-May-2014].

[34] “PostgreSQL: About,” 09-Dec-2013. [Online]. Available: http://www.postgresql.org/about/. [Accessed: 09-Dec-2013].

[35] S. Riggs and H. Krosing, *PostgreSQL 9 Administration CookBook*. 2010.

[36] C. Larman, *UML y Patrones. Introducción al análisis y diseño orientado a objeto.* Prentice Hall, 1999.

[37] “Feature Driven Development | The portal for all things FDD.” 13-Feb-2014. [Online]. Available: http://www.featuredrivendevelopment.com/. [Accessed: 13-Feb-2014].

[38] I. Sommerville, *Ingeniería del Software*, 7ma ed.

[39] E. Marcotte, *Responsive web design*. Jeffrey Zeldman, 2011.

[40] H. Hamon, “Identifyng Design Patterns in the Symfony framework.” Istanbul, Turkey, 02-May-2014.

[41] I. Gilfillan, *La biblia de MySQL*. .

[42] D. Rodríguez and R. Harrison, “Medición en la orientación a objetos,” School of Computer Science, Cybernetics & Electronic Engineering | University of Reading, UK.

[43] L. S. Almenares, “Pruebas de Carga y Estrés,” 2009.

BIBLIOGRAFÍA

G. D. Donato and G. Cefaro, *Design pattern inside Symfony*. .

E. Núñez de Shilling, “Gestión tecnológica en la empresa: definición de sus objhetivos fundamentales.,” *Revista de Ciencias Sociales (Ve)*, vol. XVII número 1, pp. 156–166, Mar-2011.

J. E. Sweat, *Guide to PHP Design Patterns. A Practical Approach to Design Patterns for the PHP 4 and PHP 5 Developer*. Canada: Marco Tabini & Associates, 2005.

J. Acosta, C. Greiner, G. Dapozo, and M. Estayno, “Medición de atributos POO en frameworks de desarrollo PHP,” Universidad Nacional del Nordeste, Argentina.

G. Booch, R. A. Maksimchuk, M. W. Engle, B. P. D. Young, J. Conallen, and K. A. Houston, *Object-Oriented Analysis and design with applications*, 3rd ed. 2007.

M. Romer, *PHP Data Persistence with Doctrine 2 ORM. Concepts, Techniques & Practical Solutions.* 2013.

S. Bergmann, *PHPUnit Manual*. 2014.

C. P. Pfleeger, *Security in Computing*, 4th ed. Prentice Hall, 2006.

Symfony. The book for Symfony2.4. 2013.