Universidad de las Ciencias Informáticas

Facultad 3

Título:

SISTEMA DE CONTROL DE MEDIOS MATERIALES DEL CEIGE

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Miguel Ferrer Oduardo

Tutor: MSc. Osmar Leyet Fernández

Co-tutor: Ing. Daileny Caridad Arias Pupo

La Habana 2013

Declaración de autoría

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

Para que así conste firmamos la presente a los ____ días del mes de _____ del año 2013.

_____

Miguel Ferrer Oduardo
Carné de Identidad: 86072223965
Autor

_____ _____

MSc. Osmar Leyet Fernández
Carné de Identidad:
Tutor

Ing. Daileny C. Arias Pupo
Carné de Identidad:
Co-tutor

" No hay tareas Imposibles, sino hombres Incapaces"

Ernesto Guevara (Che)

A mis dos mejores amigos, mis padres Dalmis y Miguel, por su apoyo incondicional, educación y guia, a ellos les debo todo lo que soy y he logrado.

A mi hermana Rosalia por sus rezos y confianza en mi.

A mis abuelas Chelo y Josefa por ser tan buena conmigo y los infinitos consejos.

A mi abuelo Raúl que a pesar de estar muerto todavia siento su cariño y dulzura.

A mi abuelo Mingo por sus palabras de consuelo y apoyo para seguir adelante en la vida.

A mis tios Raúl Luis, Adrian, Enrique, Tony, Joaquin.

A mis tias Sandra, Odalis, Berta, Gladis, Josefita, Nancy, Marian.

A mis primos y primas que los quiero mucho a todos.

A mi novia Mileidis, por su amor, correspondido.

.

Agradezco a la revolución cubana y a Fidel por brindarme la oportunidad de estudiar en esta universidad, la que considero como mi segunda casa.

De forma especial a mis tutores Osmar y Daileny, por la paciencia que tuvieron conmigo, el apoyo y los consejos, por ser mis guías en el desarrollo de este trabajo.

Al tribunal por su paciencia, guía y apoyo.

A mi oponente-cliente por su apoyo y recomendaciones.

Dioses:

Por su apoyo incondicional y disponibilidad exclusiva a Yorlen, Jean Pablo y Leo.

Semidioses:

Por su ayuda en momentos difíciles a Bartolo, Juan Luis y Amado.

Mortales:

Por sus palabras de apoyo, complicidad y amistad a mis compañeros de apto: Leiser, Sandy, Hazel, de lab: Yasmany, Leiza, Yandy, Gustavo, al club amiguitos del café: los Jorge, el Kangry, Kazuya, Ramón, Radame, Yanier, Luis Carlos, Yordan, también a Aliankys, Yolaida, Ailyn, a mis compañeros de aula a todos sin excepción.

A mis amigos de primer año con los cuales no olvido y formaron parte de una de las mejores etapa de mi vida.

En general a todos los que pudieron asistir para darme su apoyo y los que de una forma u otra ayudaron al desarrollo de este trabajo.

Gracias.

Resumen

La Universidad de las Ciencias Informáticas (UCI), creada para formar profesionales altamente calificados en la rama de la informática y producir aplicaciones y servicios informáticos, cuenta con un conjunto de centros productivos, dentro del que se encuentra el Centro de Informatización de la Gestión de Entidades (CEIGE), con el fin de desarrollar productos y servicios asociados para la gestión integral de entidades. Dicho centro actualmente no cuenta con una solución informática que gestione correctamente la información concerniente a los medios materiales que están en explotación, rotos o en almacén destinados al desarrollo de estos productos. Por lo que es necesaria la implementación de un sistema que sea capaz de gestionar dichos medios.

En tal sentido se realiza una investigación sobre los sistemas que gestionan medios materiales, tanto en el ámbito nacional como internacional, los líderes en este sentido son los sistemas de Planificación de Recursos Empresariales (ERP por sus siglas en inglés). Se valora la posibilidad de reutilizar uno de los existentes, optando por la confección de uno adecuado a las necesidades del centro, lo que conlleva al análisis de las herramientas necesarias a utilizar en su desarrollo. Se realizará una descripción de los procesos de negocio para una mejor comprensión de las tareas a automatizar, la especificación de los requisitos que debe cumplir el sistema, para ayudar en la correcta implementación de la aplicación informática. Se confeccionarán y realizarán pruebas al sistema que permitirán validar la solución propuesta. La aplicación permitirá lograr un mejor control y gestión de los medios materiales, brindando con exactitud y veracidad la información que se solicite.

Índice

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Introducción

El proceso de cambio de en el mundo ha llevado al hombre a afrontar el impacto de un nuevo milenio dentro de una sociedad que demanda cada vez más y mejores servicios para satisfacer sus necesidades. Con el rápido crecimiento de la población mundial y el desarrollo de las nuevas Tecnologías de la Informática y las Comunicaciones (TICs), han aumentado consecuentemente los recursos para llevar a cabo cualquier actividad del ser humano y de este modo facilitar su trabajo (FAO, 2009).

Debido al aumento exponencial de los recursos que demanda la sociedad se establecieron sistemas para el control y manejo de los mismos, con el fin de darles un mejor uso, distribución y evitar su escasez, permitiendo a las partes encargadas del manejo de los recursos una toma eficiente de decisiones, logrando así una mejor explotación de estos medios (FAO, 2009).

En Cuba, la existencia de resoluciones como la 60 del 2011, normas del sistema de control interno y la 249 de la responsabilidad material, exponen en su contenido cómo se deben manejar los recursos en una entidad, organismo u órgano, cuáles son los mecanismos para el correcto control de los medios, así como la aplicación de responsabilidad material. Con el apoyo de los principios básicos de control interno se garantiza que las actividades y las operaciones realizadas sobre los recursos, sean controladas y supervisadas. Se establecen las normas y procedimientos estructurados sobre la base de una adecuada organización, permitiendo evaluar su gestión de manera permanente (Consejo de Estado, 2007a).

La Universidad de las Ciencias Informáticas (UCI) como centro de altos estudios le es inherente la asignación de gran cantidad de recursos por la misión a desempeñar, en la resolución 249 encuentra el basamento legal para prevenir los daños o prejuicios y las medidas a tomar en caso de infringir alguno de ellos. Los centros productivos creados en la universidad son como pequeñas entidades, destinados al desarrollo de sistemas informáticos que tributan a determinados sectores. El Centro de Informatización de la Gestión de Entidades (CEIGE) gestiona sus medios materiales mediante hojas Excel, las cuales facilitan registrar de un medio, su estado, un número identificador, ubicación, entre otras características. Cuando existe un cambio en alguno de los elementos identificadores del medio, se crean nuevas versiones, las cuales requieren ser actualizadas en las áreas donde están ubicados estos medios. Esto demanda gran cantidad de tiempo a la persona encargada de gestionar esta actualización, pues debe generar manualmente reportes en los que el error humano puede estar presente, posibilitando que la información reflejada contenga errores.

En ocasiones no están estandarizados los nomencladores de los medios, lo que también impide que se puedan realizar conteos rápidos y obtener la cantidad de un mismo medio en un área determinada. Esta deficiencia está sujeta a que el centro maneja la información que brinda centralmente el Grupo de Activos Fijos en la universidad, no correspondiéndose la información de estos con la descripción real del medio, dificultando así el eficiente control en las revisiones que se realizan. Los reportes ofrecidos por dicho grupo presentan informaciones adicionales, las cuales no se corresponden con la necesidad actual de gestión en el centro.

Otra información que se gestiona en documentos Excel es la actualización de las partes y piezas de las computadoras, de las cuales se tienen versiones que se transmiten por correo electrónico. La actualización de los cambios de estas partes y piezas no son enviadas en tiempo real, lo que también dificulta tener la información actualizada.

La informatización y optimización de los procesos referentes a los medios materiales ofrecen disímiles ventajas como son:

- Tener información al día y sobre todo confiable, permite tomar decisiones acertadas y oportunas.
- La información pertinente de solicitudes, movimientos y reportes permite llevar un mejor control.
- La información confiable y adecuada permite definir programas de monitoreo y seguimiento a los medios del centro.

Teniendo en cuenta la situación problemática planteada anteriormente, se define como **problema a resolver:** la forma actual de gestión de la información de los medios materiales del CEIGE, presenta dificultades en cuanto a la integridad y clasificación de los datos que se generan en el proceso.

Para la misma se tendrá como **objeto de estudio:** proceso de gestión de los medios materiales.

Se identifica como **campo de acción**: sistemas informáticos para la gestión de la información de los medios materiales. Se plantea como **objetivo general:** desarrollar un sistema informático para la gestión de la información de los medios materiales del CEIGE, que contribuya a la integridad y clasificación de los datos que se generan en el proceso.

Para lograr el cumplimiento del **objetivo general,** se derivan los siguientes **objetivos específicos**:

- Elaborar el marco teórico de la investigación mediante un estudio del estado del arte de los procesos relacionados con los sistemas de control de medios materiales y las tecnologías existentes para el desarrollo de los mismos.
- Realizar el análisis y diseño de la aplicación, correspondiente con los requisitos identificados.

- Realizar la implementación del sistema mediante las herramientas y tecnologías seleccionadas.
- Validar la solución propuesta mediante pruebas de software y el cumplimiento de los objetivos de la investigación.

Para lograr el cumplimiento del **objetivo general** propuesto y solucionar la problemática planteada, se llevarán a cabo un grupo de **tareas investigativas** por **objetivos específicos** definidos:

Sobre objetivo 1:

- Definición conceptual del proceso de gestión de medios materiales.
- Valoración de los sistemas de control de medios materiales tanto en Cuba como en el mundo.
- Definición de las tecnologías a utilizar en el desarrollo de la solución de software propuesta.

Sobre objetivo 2:

- Realización del levantamiento de requisitos del sistema a desarrollar.
- Realización del diseño del sistema a desarrollar.

Sobre objetivo 3

- Realización de la implementación del sistema a desarrollar.

Sobre objetivo 4:

- Elaboración de los diseños de casos de pruebas del sistema desarrollado.
- Realización de las pruebas de aceptación al sistema desarrollado.

Idea a defender:

Si se desarrolla un sistema para la gestión de la información de los medios materiales del CEIGE, se contribuirá a la integridad y clasificación de los datos que se generan en el proceso.

Posibles resultados:

Los resultados de la presente tesis de grado fortalecerán el proceso de gestión y control de los medios materiales del CEIGE, que hoy por ser uno de los centros más grandes de la Universidad en cuanto a cantidad de personas, cantidad de proyectos y cantidad de medios materiales, es un proceso engorroso para las personas que lo realizan.

La presente tesis aportará:

- Un sistema para el control de los medios materiales del CEIGE.

El presente trabajo de diploma está estructurado en 3 capítulos:

Capítulo 1: Fundamentación teórica

En este capítulo se expone la fundamentación teórica del trabajo, incluyéndose en el mismo el estudio de otras soluciones informáticas existentes en la actualidad y de las metodologías, modelos de desarrollo de software y herramientas a utilizar.

Capítulo 2: Desarrollo de la solución

En este capítulo se realiza la propuesta de solución al problema planteado con anterioridad, exponiendo el análisis y diseño de la solución y los artefactos generados en la implementación.

Capítulo 3: Validación de la solución propuesta

En este capítulo se presentan los resultados de los casos de prueba, obteniendo una valoración integral de la solución propuesta que permita determinar los puntos débiles del sistema.

1. Capítulo 1. Fundamentación teórica

1.1. Introducción

El vertiginoso desarrollo de las TICs ha provisto al hombre de numerosas herramientas que mejoran considerablemente sus condiciones de vida y trabajo en la sociedad. Han sido varios los saltos tecnológicos y cambios de paradigmas, tanto en la telefonía, la computación como en la televisión e internet. Con todos estos cambios en el ambiente la sociedad se fue haciendo dependiente de la información y así mismo de los medios que ella exige en este siglo. Por estas razones, el uso de tecnologías adecuadamente diseñadas con especial atención a los aspectos éticos, culturales, sociales y económicos de la comunidad a la que se dirigen, cobra vital importancia en un mundo que se mueve entre bits y señales satelitales (FAO, 2009).

1.2. Gestión de medios materiales

Medios materiales

Según (ABC, 2012) los medios materiales son: aquellos bienes tangibles que permiten ofrecer los productos o servicios en cuestión.

Según (VOX LAROUSSE, 2013) los medios materiales son: medio no espiritual, ni intelectual, usado ante una necesidad para obtener lo que se pretende.

Según (Consejo de Estado, 2007a) los medios materiales son: propiedad físicamente tangible que ha de utilizarse por un período largo en las operaciones regulares de la entidad u organización y que normalmente no se destinan a la venta.

A partir de los conceptos analizados anteriormente se puede definir el concepto de medios materiales como: todos los bienes o medios que físicamente existen en una entidad u organización, destinados a la producción o servicios para acometer su rol principal, medios que pueden estar en uso o no.

Gestión de medios materiales

Según (VOX LAROUSSE, 2013) la gestión de medios materiales es: la administración, dirección o cualquier acción sobre los medios o recursos materiales para conseguir un objetivo determinado.

Según (Sonia Sánchez Andujar, 2008) la gestión de medios materiales es: función dentro de una organización que se centra en la adquisición, gestión y control de los medios materiales.

Según (Consejo de Estado, 2007b) la gestión de medios materiales es: la actividad que utiliza medios materiales permitiendo la transformación de elementos de entradas en resultados y contribuye a la eficiencia y eficacia de la empresa en el logro de sus objetivos.

A partir de los conceptos analizados anteriormente se puede definir el concepto de gestión de medios materiales como: todas las acciones realizadas sobre dichos medios, por la entidad a la que pertenecen o sus responsables directos.

También se pueden llamar medios o recursos materiales a: instrumentos, útiles y herramientas, partes y piezas de repuesto, producciones terminadas, etc. En fin todos aquellos medios que representen valor dentro de una entidad, destinados al consumo o comercialización de ella (Consejo de Estado, 2007a).

Una entidad, institución o centro que gestiona y controla correctamente los medios a su disposición se ve involucrada en varios procesos, entre los más importantes se pueden mencionar movimientos de medios, altas y bajas de medios, solicitudes de medios, cambios de estados de los medios, etc.

1.3. Soluciones informáticas en la gestión de medios materiales

La gestión de los medios materiales a todos los niveles, va cobrando cada día mayor importancia. La necesidad de aprovechar al máximo su potencial, se ha convertido en una necesidad en estos tiempos, motivo por el cual resulta imperioso el uso de soluciones informáticas para la gestión de estos medios. La ventaja más significativa es que permiten a las empresas automatizar muchos aspectos de la gestión de medios materiales, logrando reducir de este modo la carga de trabajo del departamento encargado del control de dichos medios. Al mismo tiempo, permite, aumentar la eficiencia del departamento a través de la estandarización de sus procesos.

Contar con el debido control de los medios materiales es un elemento clave para la gestión de las organizaciones. Es la administración la que debe velar por estos medios y su manejo, tratando de llevar su explotación de forma óptima.

Los sistemas que dedican parte de sus procesos a la gestión y control de recursos materiales son los (ERP por sus siglas en inglés) para Planificación de Recursos Empresariales. Estos sistemas están compuestos por varios módulos, entre ellos algunos para el control de los activos fijos tangibles o medios materiales. El principal objetivo de estos sistemas es ayudar a las empresas como un todo y en particular este módulo contribuye con el departamento de recursos materiales.

El análisis de los datos obtenidos de estos sistemas provee a la organización de información veraz y muy valiosa, lo que permite mejorar la capacidad de toma de decisiones de los directivos. A continuación se describen algunas de las herramientas de interés para la investigación, ya sea por su notoriedad en el mercado o por las características que las relacionan con el contexto del problema planteado.

1.3.1. Soluciones internacionales

Oracle JD Edwards EnterpriseOne

Es un conjunto completo de aplicaciones comerciales modulares, previamente integradas, específicas del sector, diseñadas para una rápida implementación y una fácil administración. Es adecuado para las organizaciones que fabrican, construyen, distribuyen, brindan servicios o administran productos o activos físicos. Diseñado para obtener un costo superior de propiedad con funcionalidad totalmente incorporada en un solo grupo de herramientas, un solo modelo de datos y una sola interfaz de usuario.

El sistema Activos Fijos de J.D. Edwards proporciona una forma eficaz de dar seguimiento a los medios a fin de satisfacer las necesidades financieras y de elaboración de informes (Oracle, 2012).

Entre otras funcionalidades permite:

- Producir informes de medios.
- Conciliar medios.

Openbravo

Es un sistema de gestión empresarial integrado de software libre y entorno web, que ofrece una propuesta de valor radicalmente distinta, proporcionando mucho más por mucho menos. Este sistema está orientado a la gestión empresarial integrada dentro de pequeñas y medianas empresas (PYMES), permitiendo la administración y optimización del sistema productivo, lo que repercute en un aumento de la eficiencia de la firma.

Openbravo está licenciado bajo Openbravo Public License Version 1.1 ("OBPL") que es una adaptación de la licencia libre Mozilla Public License (MPL), licencia que cumple completamente con la definición de software de código abierto de la Open Source Initiative (OSI) y con las cuatro libertades del software libre enunciadas por la Free Software Foundation (FSF) (ANON, 1999).

El sistema de gestión Openbravo agrupa en grandes áreas cada uno de los componentes de la empresa, lo que permite llevar un control más estricto, aunque el programa presenta diferentes módulos y submódulos siempre es sencillo llegar al destino necesario (ACOSTA, MEDINA, 2007).

Entre los submódulos se encuentran: plan de cuentas, cuentas contables, presupuestos, impuestos, contabilidad general, cuentas a pagar, cuentas a cobrar, contabilidad bancaria, balance, cuenta de resultados, activos fijos.

El submódulo de activos fijos posibilita:

- Definición de grupos de activos.
- Permite definir los activos de la empresa.
- Permite clasificar los activos en distintas categorías. Cada categoría se define por un nombre.

Vtiger

Es un sistema que ayuda a las empresas a gestionar su ciclo de venta y las relaciones con los clientes, este sistema para la gestión de las relaciones con los clientes (Customer Relationship Management) CRM Vtiger, es de código abierto, está compuesto por cuatro módulos que funcionan como un todo, entre ellos se encuentra el de Inventario el cual aporta a la solución características novedosas en cuanto a productos compuestos. Esta característica permite construir un orden jerárquico en los que hay subproductos relacionados con un producto paterno. Un producto puede ser un subproducto de un número ilimitado de productos, pero un producto paterno nunca puede convertirse en un subproducto (Vtiger, 2012). Esta relación es la deseada para obtener un mejor control de la solución a desarrollar en las partes y piezas de una computadora, con dicha computadora como padre de estas partes y piezas.

Assets

ASSETS NS es un Sistema de Gestión Integral estándar que permite el control de los procesos de Compras, Ventas, Producción, Taller, Inventario, Finanzas, Contabilidad, Presupuesto, Activos fijos, Útiles y Herramientas y Recursos humanos. Como sistema integral todos sus módulos trabajan en estrecha relación, generando, automáticamente, al módulo de Contabilidad los comprobantes de operaciones por cada una de las transacciones efectuadas, esto permite que se pueda trabajar bajo el principio de Contabilidad al Día (ASSETS, 2013).

Es un sistema flexible, amigable, con ayuda en línea que puede ser instalado en una microcomputadora o sobre varias, funcionando en ambiente multiusuario incluidas estaciones remotas. Asimismo, proporciona opciones de seguridad que le permiten limitar el acceso a los diferentes procesos del sistema de acuerdo con el perfil de cada usuario(ASSETS, 2013).

Las opciones que brinda este módulo Activos fijos son las siguientes:

- Entradas de activos fijos.
- Salidas de activos fijos.
- Movimientos de activos fijos.
- Conteo físico de activos fijos.
- Listados de activos fijos.

1.3.2. Soluciones nacionales

RODAS XXI

Fue desarrollado por la empresa CITMATEL, el Sistema Integral Económico Administrativo RODAS XXI permite automatizar el funcionamiento de cualquier empresa o unidad presupuestada. Desarrollado con los más recientes adelantos en el diseño de interfaz de usuario, es un sistema en constante desarrollo que tiene muy en cuenta su opinión para perfeccionarse y le ofrece cada vez mejores soluciones que harán más viable y rápido su trabajo. Es un sistema multiempresa que cuenta actualmente con seis módulos: Finanzas, Contabilidad, Activos fijos, Nóminas, Inventario y Facturación. Estos módulos pueden emplearse integrados en su totalidad, formando cualquier subconjunto entre ellos, o cada uno de forma independiente.

El módulo de activos fijos hace un control detallado de los activos fijos de su entidad. Se pueden realizar todo tipo de operaciones de activos fijos con facilidad en el momento que se desee, generando el documento asociado a cada acción.

El módulo de Activos fijos de RODAS XXI destaca por ser sencillo y fácil para el trabajo en él, permitiendo así un rápido aprendizaje, acortando el tiempo necesario para que nuevos usuarios se les facilite el trabajo con el módulo (ANON, 2002).

Versat-Sarasola

Es desarrollado por la casa de software TEICO Villa Clara. Está programado en Delphi y SQL Server 2000. Su nombre viene de la palabra Versatilidad (Versat) y Sarasola del apellido de un eminente contador cubano. Está implantado en más de 2500 unidades presupuestadas del país, entre las que figuran organismos de la Administración Central del Estado, las direcciones municipales de finanzas, tesorerías, la ONAT y otros.

El Versat cuenta con los módulos de Configuración, Contabilidad general y Costos, como elementos básicos e imprescindibles del paquete. Además contiene los módulos adicionales de Finanzas, Inventario, Activos fijos y Facturación. La interrelación entre estos subsistemas económicos se logra a partir del mecanismo de contabilización implementado, donde todos los subsistemas tributan comprobantes para la contabilidad, la cual los valida y los asienta en los libros.

Control de Activos Fijos: Recoge las operaciones que se realizan sobre los medios, las más significativas son: altas, bajas y modificaciones. (MSc.Porteiro, Lic.Morales, 2008).

ARCE

Proyecto ARCE (Aplicación de Red para Casos de Emergencia), es una aplicación dirigida a la protección civil que tipifica el intercambio de los recursos y medios, consultas, apoyos logísticos, entre otras gestiones. Este es un proyecto de colaboración iberoamericana a la que pertenecen Argentina, Bolivia, Brasil, Chile, Colombia, Costa Rica, Cuba, Ecuador, El salvador, España, Guatemala, Honduras, México, Nicaragua, Venezuela, entre otros países (Santiago Escárez San Martín, 2008). Este sistema permite una adecuada catalogación, gestión de los inventarios de medios y recursos y organiza la información de forma tal que el acceso a ella sea rápido. El objetivo del catálogo es conocer, organizar y localizar en forma rápida y eficaz los medios y recursos. La codificación de la catalogación está definida por ARCE, esto permite obtener informes, controlar y gestionar la base de datos. Entre otros datos de los recursos y medios se controlan titularidad, localización y sector. El mayor inconveniente de esta aplicación es que se debe realizar una inversión de más de $ 200.000 dólares para poder adquirirla.

Cedrux

Es una solución genérica para la gestión de entidades basada en los principios de independencia tecnológica. El sistema automatiza funcionalidades generales de los procesos que tienen lugar en los organismos, teniendo en cuenta las particularidades de la economía cubana. Entre sus principales beneficios está la sustitución de importaciones por conceptos de sistemas informáticos de gestión empresarial, además de que servirá como herramienta fundamental para el control de los medios del país y contribuirá a evitar pérdidas por desvíos y robos. Pone al servicio de la entidad las potencialidades de la tecnología informática y provee facilidades para la integración de las diferentes áreas productivas y departamentos administrativos. Las características generales a tener en cuenta de la solución en referencia al módulo de activos fijos son:

- Traslado de activo a ocioso.
- Movimientos de activos.
- Alta de activo fijo tangible.
- Baja de activo fijo tangible.
- Traslados internos de activos fijos tangibles.

Destacar que aunque muchos de los procesos que se llevan a cabo en esta solución son ideales para el sistema a implementar, este módulo se encuentra todavía en producción.

Gespro

Solución desarrollada en la UCI, es una herramienta para la gestión de proyecto (GESPRO) actualmente en su versión 12.5, la cual "permite resolver cualquier tipo de incidencia de forma ordenada, rápida y eficiente, logrando una mayor calidad en el servicio de soporte" (ANON, 2006).

La gestión que brinda el Gespro es a través de un plugin llamado **Recursos**; el control de los recursos se ve en dos niveles: administrativo y a nivel de proyecto. Permiten:

- Gestionar categoría
- Gestionar recursos
- Solicitar recursos

1.3.3. Resultado del estudio de las soluciones informáticas

Las soluciones informáticas estudiadas en este capítulo, responden a las necesidades de las entidades para las que fueron desarrolladas, por lo que no son lo suficientemente adaptables para satisfacer las necesidades planteadas en la problemática de esta investigación. Las características especificadas de los módulos que contribuyen al control de activos, medios o recursos de cada solución vista, son las que le aportan al sistema funcionalidades novedosas.

El estudio de los diferentes sistemas arrojó varias propuestas que pueden ser utilizadas en la aplicación, la "X" representa el aporte en referencia a procesos o componentes de una herramienta. Mediante la técnica de entrevista al cliente se identificaron como procesos principales los reflejados en la tabla, los sistemas representados en dicha tabla son los que aportan una práctica más funcional para dichos procesos o componentes.

Procesos o Componentes	Versat-Sarasola	JDEdwards	Rodas XXI	Openbravo	Assets

Cambio de estado	**x**	**x**			**x**
Movimientos		**x**	**x**	**x**	**x**
Generación de documentos		**x**	**x**	**x**	**x**
Modificaciones de los medios	**x**	**x**			**x**

Tabla 1: Aporte de las soluciones más relevantes.

La implantación de uno de estos sistemas estudiados o de características similares, lejos de solucionar el problema existente, trae consigo la adquisición de una tecnología costosa y de alto rendimiento, lo cual no sería factible, ya que la cantidad de medios a gestionar no es tan grande. La obtención de otros módulos cuando solo se necesita el de Activo Fijos Tangibles (AFT), Inventario o el referente al control de medios y recursos, sería un gasto innecesario. En caso que este módulo pueda ser instalado independiente, los procesos involucrados en él exceden el control que se quiere realizar sobre los medios del centro, por lo tanto, se tendrían procesos que no serían utilizados. Por otra parte, existen en los sistemas estudiados características que sirven de apoyo para el desarrollo de esta nueva solución, dichas características se encuentran en las interfaces mostradas al cliente, los datos que se manejan en ellas, la forma en que tratan los conceptos, la apariencia que debe tener un sistema de este tipo, etc.

Por lo anteriormente expuesto se concluye que:

- Las soluciones informáticas estudiadas no suplen las necesidades específicas del problema planteado.
- La mayoría de estas soluciones son privativas o emplean tecnologías privadas.
- La forma de controlar los medios no se ajusta a las necesidades del centro.

La solución debe ser una aplicación web, por las bondades que brinda el uso de la arquitectura cliente-servidor y deben utilizarse herramientas de software libre para su desarrollo.

1.4. Proceso de desarrollo de software

El proceso de desarrollo de software "es aquel en que las necesidades del usuario son traducidas en requisitos de software, estos requisitos transformados en diseño y el diseño implementado en código, el código es probado, documentado y certificado para su uso operativo". Concretamente "es una estructura aplicada al desarrollo de un producto software que define quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo" (Jacobson, 1998).

El proceso de desarrollo de software requiere por un lado un conjunto de conceptos, una metodología y un lenguaje propio. A este proceso también se le llama el ciclo de vida del software que comprende cuatro grandes fases: concepción, elaboración, construcción y transición (Zavala Ruiz, 2008).

Para lograr la consecución exitosa de un proyecto de software, existen un conjunto de modelos que permiten perfilar el proceso de desarrollo, los cuales deben ser analizadas cuidadosamente. Usualmente la selección de uno de estos modelos o de la unión de varios responde a las necesidades y condiciones específicas del proyecto.

Estos modelos o metodologías se pueden clasificar en cinco grandes grupos (Software Development Methodologies, 2012):

- **Modelo de cascada**: provee un enfoque muy lineal para el desarrollo de software, ya que cada paso en esencia lleva al siguiente hasta que todo el software se ha completado.
- **Modelo iterativo incremental**: en muchos aspectos, el modelo incremental emula el modelo de cascada, con la diferencia que recomienda la construcción de secciones reducidas de software que irán ganando en tamaño para facilitar así la detección de problemas de importancia antes de que sea demasiado tarde.
- **Modelo espiral**: en cierto modo, el modelo de espiral es como una versión más grande e intensa del modelo incremental. Este modelo también es conocido por permitir el desarrollo de software de alta calidad, este resultado es obtenido gracias a la característica fundamental del modelo que es la gestión de riesgos de forma periódica en el ciclo de desarrollo.
- **Modelos ágiles**: en la gestión de proyectos ágiles, existe un esqueleto básico conformado con las mejores prácticas basadas en el sentido común, pero la gestión global se basa en otros métodos. Este modelo utiliza un desarrollo iterativo como base para abogar por un punto de vista más ligero y más centrado en las personas. Los procesos ágiles utilizan retroalimentación en lugar de planificación como principal mecanismo de control.
- **Modelo de prototipos**: en el método de creación de prototipos, el software es desarrollado en forma similar a una cebolla, o sea, en capas. Cuando el proyecto se inicia, el equipo se centra en la construcción de un prototipo con todos o la mayoría de las características y construye todo un programa para que el cliente lo utilice. Después de que el cliente acepta el prototipo, el equipo de programación crea la siguiente capa mediante la corrección de los errores y basándose en los requisitos fundamentales.

Teniendo en cuenta los modelos anteriormente planteados y las características de la solución que se desea desarrollar, de ser un sistema pequeño, con un equipo de trabajo que no exceden las cinco personas. Los modelos ágiles, junto a los modelos de prototipos, proveen un marco idóneo para el desarrollo del producto, debido a que estos tipos de modelos ofrecen las siguientes ventajas (Sousa, 2009):

- 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.
- Debido a la retroalimentación constante, se hace más fácil hacer frente a 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.

Entre las principales metodologías ágiles se puede encontrar (Appelo, 2008):

- **Scrum:** es una metodología de gestión de proyectos ágil, creado por Ken Schwaber y Sutherland Jeff. Se trata de un esqueleto que incluye un pequeño conjunto de prácticas y roles predefinidos. Para la gestión de proyectos ágiles, Scrum se está convirtiendo en una metodología de desarrollo de software de mucha popularidad. Una de las razones es que Scrum sólo consta de ciertas prácticas de sentido común que pueden aplicarse en muchas situaciones. Esto también significa que Scrum por sí mismo nunca es suficiente, y que los equipos de desarrollo tienen que utilizar simultáneamente otros métodos (por lo general XP) para prácticas adicionales.
- **Extreme Programming (XP):** es una metodología de ingeniería de software ágil, creado por Kent Beck. Se trata de un conjunto de buenas prácticas de los cuales algunos son llevados a niveles “extremos”. Al igual que con otros métodos ágiles, XP se refiere a los cambios en curso como un

aspecto natural y deseable en el desarrollo de software. XP es a menudo visto como un complemento a Scrum, llenando la mayor parte de los agujeros que deja abiertos Scrum.

- **Agile Unified Process:** (AUP por sus siglas en inglés), es una versión simplificada del Proceso Unificado Relacional (RUP por sus siglas en inglés). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. Abarca siete flujos de trabajos, cuatro ingenieriles y tres de apoyo: Modelado, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyectos y Ambiente.

1.5. Lenguajes y herramientas de modelado

Usualmente para desarrollar una solución informática con calidad y competitiva, el uso de lenguajes de modelado es prácticamente indispensable. Esta necesidad responde a que el uso de estos lenguajes permite que el desarrollo de software se formalice a través de estándares unificados. Se entiende por lenguaje de modelado cualquier lenguaje artificial que puede ser utilizado para expresar la información, el conocimiento o sistemas en una estructura que está definida por un conjunto coherente de reglas. Explicar el modelo de negocio, es la parte difícil de hacer llegar al cliente la solución propuesta a un problema planteado. Están a disposición de los usuarios una amplia variedad de herramientas para ayudar a obtener una clara imagen de desempeño y rendimiento del negocio a desarrollar: hojas de cálculo, tablas, gráficos, resultados operativos, reportes de estado, etc. Estas herramientas permiten a los analistas junto a los clientes realizar análisis *"y si"* y analizar preguntas como *"¿Cómo puedo realizar…?"*. Las reglas se utilizan para la interpretación del significado de los componentes en la estructura. Los lenguajes de modelado utilizan modelos visuales y diagramas que realizan esa representación de manera precisa, entendible y económica, lo que facilita su uso en las herramientas CASE convencionales, un ejemplo de esto es la integración que existe entre al Visual Paradigm y el Lenguaje Unificado de Modelado (UML). (Zapata, 2009)

Estos lenguajes se clasifican en dos tipos:

- Lenguajes gráficos: utiliza una técnica de diagramas con símbolos que representan los conceptos mencionados y las líneas que conectan los símbolos representan las relaciones.
- Lenguajes textuales: suelen utilizar palabras clave estandarizadas acompañadas de parámetros para que las expresiones sean interpretables por un ordenador.

Notación de modelado de proceso de negocio

Un modelo de proceso de negocio consiste en un conjunto de modelos de actividades y las limitaciones de ejecución entre ellos. Una instancia de proceso de negocios representa un caso concreto en el negocio operativo de una empresa, que consiste en las instancias de cada actividad. Cada modelo de procesos de negocio actúa como un modelo para un conjunto de instancias de procesos de negocio, y a la vez cada actividad actúa como modelo para un conjunto de instancias de actividad (Weske, 2007).

Notación de Modelado de Proceso de Negocio (BPMN por sus siglas en inglés), es el estándar más reciente para modelado de procesos del negocio y servicios. BPMN es una notación necesaria para expresar los procesos de negocio en un único diagrama de proceso de negocio, BPMN permite hacer un mejor uso de la gestión de procesos del negocio (BPM por sus siglas en inglés), ya que normaliza el método de notación que sirve como ayuda en la automatización de los procesos (Fajardo Ugarte, 2008).

El principal objetivo de BPMN es proporcionar una notación estándar fácilmente comprensible por todos los miembros de la empresa. Por lo tanto, BPMN sirve como un lenguaje común, reduciendo la brecha de comunicación que se produce con frecuencia entre el diseño de procesos de negocio y la implementación.

Lenguaje unificado de modelado

Unified Modeling Language (UML) es un leguaje de modelado 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 (Pressman, 2004).

Es importante resaltar que UML es un "lenguaje" para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir.

Los estándares que propone el lenguaje UML ofrecen innumerables ventajas para el proyecto. Teniendo en cuenta que la solución que se desea desarrollar se desenvuelve en un dominio, resulta más complicada la representación a través de procesos de negocio, por lo que el uso de UML se hace indispensable.

Visual Paradigm

Visual Paradigm es una herramienta de Ingeniería de Software Asistida por Computadoras (CASE por sus siglas en inglés). 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 (Visual Paradigm, 2012).

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 diseñado 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 (Visual Paradigm, 2012).

- Entorno de creación de diagramas para UML.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Generación de bases de datos.
- Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Exporta los diagramas como imágenes (.jpg) y otros formatos.

1.6. Ambiente de desarrollo

A continuación se describirán lenguajes, librerías, herramientas y frameworks que pueden ser utilizados en el desarrollo de la solución.

1.6.1. Lenguajes de programación

Un lenguaje de programación es un idioma artificial diseñado para expresar procesos que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina o para expresar algoritmos con precisión. A continuación se describen algunos de los lenguajes de programación más populares y acordes a las necesidades de la solución informática que se desea desarrollar (Microsoft, 2012).

PHP

PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. PHP es ampliamente utilizado para fines generales aunque es especialmente adecuado para el desarrollo web y puede ser embebido en páginas HTML (The PHP Group, 2012).

PHP brinda la posibilidad de usar programación procedimental o programación orientada a objetos (POO), o una mezcla de ambos. Una de las características más fuertes y más importantes de PHP es su soporte para una amplia gama de bases de datos. Escribir una página web habilitada para la base de datos es muy simple utilizando una de las extensiones de bases de datos específicas (por ejemplo, para MySQL), o el uso de una capa de abstracción como Objetos de Datos PHP (PDO por sus siglas en inglés), o conectarse a cualquier base de datos que soporte el estándar de base de datos de conexión abierta a través de la extensión de Conectividad Abierta de Base de Datos (ODBC por sus siglas en inglés).

Como se puede apreciar, son innumerables las características y beneficios que PHP ofrece, principalmente a las aplicaciones web, como es el caso de la solución informática que se desea desarrollar. Si a ello se agrega la existencia de una amplia comunidad formada por expertos programadores y la disponibilidad de frameworks de desarrollo que facilitan el trabajo con el lenguaje, PHP se convierte en una excelente opción para llevar a feliz término la aplicación. La versión 5.3 del lenguaje provee características que mejoran el rendimiento y seguridad del software, permitiendo de esta manera una eficiente gestión de la información.

Java

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. Java puede ser considerado un lenguaje de propósito general, pues es utilizado en el desarrollo de aplicaciones de escritorio, pasando por aplicaciones web, hasta las aplicaciones para teléfonos móviles. Su sintaxis es fácil de usar y está soportado por una amplia comunidad de programadores expertos.

Java utiliza una máquina virtual (bytecode) para interpretar y traducir las instrucciones de alto nivel a lenguaje binario entendible para las computadoras. El uso de este intermediario supone un uso menos eficiente de la memoria, pero suprime gran cantidad de errores originados por la incorrecta manipulación de punteros.

Entre noviembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de acuerdo con las especificaciones del Java Community Process, todo el Java de Sun es ahora software libre.

1.6.2. Librerías y frameworks

Un framework simplifica considerablemente el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Proporciona una estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener, además de facilitar la programación de aplicaciones pues encapsula operaciones complejas en instrucciones sencillas. Por otro lado las librerías persiguen un objetivo similar, con la diferencia de que están orientadas a un uso más pasivo y usualmente no generan código fuente en la aplicación sino que el desarrollador hace uso de las funcionalidades que esta provee. A continuación se describen algunos de los principales frameworks de desarrollo para PHP y librerías Javascript.

Symfony

Symfony es un framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web (Potencier, 2008).

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas Unix, como en plataformas Windows. A continuación se muestran algunas de sus características (Potencier, 2008).

- Fácil de instalar y configurar en la mayoría de plataformas.
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.

- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

Symfony constituye una perfecta elección para el desarrollo de la solución informática que se desea desarrollar, hecho que responde a las características y buenas prácticas de diseño que hacen de este framework líder indiscutible en la implementación de aplicaciones web con PHP.

Symfony provee características y filosofías fundamentales para el desarrollo de una plataforma, de la cual la solución que se desea desarrollar constituye un núcleo base de vital importancia. Si a ellos le sumamos la existencia de gran cantidad de plugins y de una numerosa y activa comunidad, Symfony se convierte en un excelente opción para desarrollar este tipo de aplicaciones.

Zend Framework

Zend Framework es un marco de trabajo que se basa en la simplicidad, está completamente orientado a objetos gracias al uso de buenas prácticas y posee una base de código rigurosamente ágil. Zend Framework se centra en la construcción de soluciones webs seguras, confiables y modernas. También es utilizado extensivamente en la creación de servicios web y APIs (Zend Technologies Ltd, 2012).

Ofrece un gran rendimiento y una robusta implementación del Patrón Modelo Vista Controlador, una abstracción de base de datos fácil de usar, y un componente de formularios que implementa la prestación de formularios HTML, validación y filtrado para que los desarrolladores puedan consolidar todas las operaciones usando de una manera sencilla la interfaz orientada a objetos (Zend Technologies Ltd, 2012).

JQuery

JQuery es una librería de JavaScript rápida y concisa que simplifica el uso del documento HTML, el manejo de eventos, la animación, y las interacciones Ajax para el desarrollo web rápido (The jQuery Project, 2010).

Anunciado en 2006 por su creador, John Resig, jQuery ganó rápidamente popularidad y apoyo al convertirse en una nueva forma de utilizar JavaScript para interactuar con HTML y CSS. Los selectores de jQuery son simples, imitando selectores CSS, por lo que es una biblioteca familiar y fácil de aprender para los diseñadores y desarrolladores por igual. La librería jQuery borró la preocupación que los desarrolladores web habían sufrido al intentar crear sitios interactivos a través de una amplia gama de navegadores por el manejo de la mayoría de los problemas de compatibilidad de navegadores. Aparte de ser muy fácil de usar, una de las mayores ventajas de jQuery es que maneja una gran cantidad de

"estándares" para varios navegadores web. En la actualidad esta librería es ampliamente utilizada en disímiles aplicaciones web. Su reputación ha crecido exponencialmente al igual que la comunidad de desarrolladores que comparten sus experiencias así como plugins que complementan el trabajo con la librería (Castledine, Earle, Sharkie, 2010).
La última versión estable de jQuery es la 1.7.2, la cual es seleccionada para ser utilizada en el desarrollo de la solución informática propuesta.

Ext JS

Ext JS es una librería JavaScript ligera y de alto rendimiento, que permite crear páginas e interfaces web dinámicas. Entre sus principales ventajas se encuentra el balance entre Cliente-Servidor, la carga de procesamiento se distribuye permitiendo que el servidor, al tener menor carga, pueda manejar los clientes de manera más eficiente, además su fuerte paradigma basado en componentes soportado por recursos para la programación orientada a objetos en Java Script que facilitan la implementación de extensiones y aplicaciones de gran complejidad, el poseer un amplio conjunto de componentes configurables de alta calidad y una implementación transparente. La comunicación asíncrona permite el intercambio de información con el servidor sin necesidad de estar sujeta a una acción del usuario, dando la libertad de cargar la información sin que este lo note (Jesús García, 2011).

1.6.3. Herramientas de desarrollo

A continuación se abordan las herramientas de desarrollo que pueden ser usadas como apoyo en la implementación de la solución.

Eclipse

Eclipse es un Entorno de Desarrollo Integrado (IDE), de código abierto y multiplataforma. Es una potente y completa plataforma de programación, desarrollo y compilación de elementos tan variados como sitios web, programas en C++ o aplicaciones Java.
Eclipse, al mismo tiempo, es una comunidad de código abierto, cuyos proyectos se centran en la construcción de una plataforma de desarrollo abierta compuesta por marcos extensibles, herramientas de tiempos de ejecución para la construcción, implementación y administración de software a través del ciclo de vida de los mismos. La Fundación Eclipse es una organización sin fines de lucro que aloja los

proyectos de Eclipse y ayuda a cultivar tanto una comunidad de código abierto y un ecosistema de productos y servicios complementarios (Eclipse Foundation, 2012).

NetBeans

NetBeans es un entorno de desarrollo integrado 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 (Oracle Corporation, 2012).

NetBeans es una de las opciones para el desarrollo con Java, que recibe soporte integrado para lenguajes dinámicos, todo en una herramienta de gran alcance. El editor de PHP de NetBeans ofrece plantillas de código y generación (getters y setters), la refactorización, información sobre herramientas de parámetros, consejos, soluciones rápidas y la finalización de código inteligente. Provee de un resaltado de código sintáctico y semántico, documentación emergente, formateo de código y plegado, así como el marcado de los sucesos y puntos de salida.

El IDE también ofrece un completo editor de HTML, JavaScript y CSS, con la ventaja de proveer un resaltado de sintaxis completa, completamiento de código inteligente, documentación pop up, y la comprobación de errores para HTML, CSS y JavaScript, incluyendo HTML 5, JavaScript 1.7. El editor reconoce código HTML en los archivos de JavaScript y viceversa. El editor reconoce también HTML y JavaScript en XHTML, PHP y archivos JSP.

1.7. Sistema gestor de base de datos

Un sistema gestor de bases de datos o administrador de base de datos, es un programa que permite a uno o más usuarios crear y acceder a los datos en una base de datos. Los sistemas gestores de base de datos, gestionan las peticiones del usuario para que estos sean libres de entender que los datos se encuentran físicamente en los medios de almacenamiento. En el manejo de solicitudes de los usuarios, asegura la integridad de los datos asegurándose de que sigue siendo accesible y no sufra cambios, y que sólo aquellos con privilegios de acceso pueden acceder a los datos. El más típico DBMS (Data Base Management System), son las siglas en inglés para los Sistemas de Gestión de Bases de Datos (SGBD) es un sistema de base de datos relacional (Christiansen, Simon, 2012).

La selección adecuada del DBMS proveerá a la solución informática que se desea desarrollar numerosas herramientas, además de seguridad a los datos que se almacenan y mejorará considerablemente el

desempeño de la aplicación final. A continuación se describen algunos de los SGBD más utilizados en software con características similares.

MySQL

MySQL, es el más popular sistema de gestión de base de datos multiplataforma de código abierto. Es desarrollado, distribuido y soportado por Oracle Corporation. Fue creado en 1995 por David Axmark, Allan Larsson y Michael Widenius (Oracle Corporation, 2012).

MySQL se caracteriza fundamentalmente por su velocidad, rendimiento y fiabilidad, motivo por el cual es ampliamente utilizado en sitios web y aplicaciones de escritorio de baja y mediana complejidad. Muchas de las organizaciones más grandes y de más rápido crecimiento del mundo, incluyendo Facebook, Google, Adobe, Alcatel Lucent y Zappos basan sus sistemas en MySQL (Oracle Corporation, 2012).

Tanto el software del servidor de MySQL en sí y las bibliotecas de cliente utilizan una doble licencia de distribución. Se ofrecen bajo licencia GPL, a partir del 28 de junio de 2000 (que Oracle ha ampliado con una excepción de licencia de software libre) (Oracle Corporation, 2012) o una licencia propietaria. Precisamente por esta razón el uso de MySQL no es recomendado para la solución informática propuesta, puesto que Oracle puede retirar en cualquier momento la licencia GPL del DBMS y se estarían incumpliendo las políticas de soberanía tecnológica de la universidad y del país.

PostgreSQL

PostgreSQL es un avanzado servidor de base de datos SQL, disponible en una amplia gama de plataformas. Además de ser una herramienta de código abierto, las versiones de PostgreSQL son mantenidas por largos períodos de tiempo y requiere poco o ningún mantenimiento (Riggs, Simon, Krosign, Hannu, 2010).

Entre las características fundamentales de PostgreSQL se pueden encontrar (Riggs, Simon, Krosign, Hannu, 2010):

- Excelente cumplimiento de los estándares SQL 2008.
- Arquitectura cliente-servidor.
- Diseño altamente concurrentes donde los lectores y escritores no se bloquean entre sí.
- Altamente configurable y extensible para muchos tipos de aplicaciones.
- Integridad referencial (ForeignKeys).
- Triggers o disparadores.

- Integridad Transaccional (ACID).
- Control de versionado concurrente (MVCC).
- Tiene licencia de tipo BSD.

PostgreSQL se caracteriza además por ser un sistema de base de datos de propósito general que permite crear sus propias funciones de servidor en una docena de idiomas diferentes. Los sistemas gestores de base de datos son altamente extensibles, de modo tal que es posible agregar sus propios tipos de datos, operadores y los tipos de índices.

1.8. Conclusiones

En el capítulo se realizó un estudio de los principales conceptos asociados a la gestión de los medios materiales, permitiendo lograr una mejor comprensión de los conceptos asociados al sistema. Además se realizó una valoración de los sistemas que dedican todos o algunos de sus procesos al control de medios materiales, obteniendo características que apoyarán el desarrollo de la solución propuesta. Con la investigación y estudio de las tecnologías, librerías y herramientas se realizó una selección de las más indicadas para el desarrollo del sistema. Por las características de la solución a desarrollar de ser un proyecto pequeño, con un reducido equipo de trabajo y realizar pequeñas entregas funcionales, se escoge como metodología de desarrollo ágil XP, que permite generar los artefactos necesarios en la fase de implementación, es flexible en la exigencia de la cantidad de diagramas y artefactos a generar, esto es posible ya que el cliente es parte del equipo de trabajo y está relacionado estrechamente con su desarrollo, además todas sus facilidades van orientadas al programador. El modelado se realizará con UML en su versión 2.0 lo cual permitirá construir, visualizar y documentar los artefactos del sistema; se seleccionó como herramienta CASE Visual Paradigm en su versión 8.0 por la integración que posee con UML y por la ventaja que ofrece a la hora de describir los procesos y crear artefactos, Visual Paradigm acompaña todo el ciclo de vida del proyecto generando todo tipo de diagramas. El framework de desarrollo escogido es Symfony2, en esta versión Symfony hace uso de las mejores prácticas existentes en la actualidad para el desarrollo web, fácil de instalar e integrar con librerías que ayudan a la construcción del sistema. Para la creación de las interfaces se seleccionó jQuery que es una librería JavaScript liviana y concisa, una de sus mayores ventajas es que maneja una gran cantidad de estándares para varios navegadores web. Como sistema gestor de base de datos se escoge PostgreSQL en su versión 9.1, como IDE el NetBeans en la versión 7.3 orientado a la programación web. Como parte

de la definición propuesta para la solución, se decide el uso de herramientas libres acorde con las políticas tecnológicas de la universidad y del país.

2. Capítulo 2: Desarrollo de la solución

2.1. Introducción

En este capítulo se tiene como objetivo fundamental la presentación de la solución propuesta, así como brindar los diseños pertinentes para una mejor comprensión de las funcionalidades que se implementarán. Se muestran en el contenido del acápite las principales clases identificadas, se describen algunos flujos significativos para el desarrollo del software, diagramas, patrones utilizados, entre otras especificaciones que ayudarán en el desarrollo de la solución.

2.2. Descripción de la propuesta de solución

Siguiendo el estudio del capítulo anterior y teniendo presente los objetivos de este trabajo de diploma, se propone un sistema para el control y gestión de los medios materiales del CEIGE.

La solución constará una base de datos de todos los medios que se encuentran en el centro. La duplicación de la información provee la ventaja de que se podrá guardar en el centro los históricos de todas las acciones que se lleven a cabo con los medios, lo cual actualmente no existe.

La aplicación debe permitir hacer búsquedas según los criterios establecidos, conteos de los medios, además de generar reportes, teniendo en cuenta el rol y nivel de acceso de los usuarios de la aplicación, garantizando así la integridad y disponibilidad de los datos. En el mismo sentido la aplicación brindará la posibilidad de gestionar los medios, ayudando a una mejor organización y confidencialidad de la información.

2.3. Descripción de los procesos de negocio

La descripción de los procesos de negocio ayuda a comprender gráficamente como se realizan los procedimientos dentro de las entidades, en el caso de la metodología escogida esta descripción se realiza en forma de párrafo, de forma sencilla y concisa, utilizando un lenguaje natural fácilmente entendible por los clientes y el equipo de trabajo. En el caso de esta investigación la descripción se realizará sobre una plantilla, la cual está definida por el centro para ello, de esta forma contribuirá a un mejor entendimiento.

Objetivo	Definir el tratamiento a seguir en los casos de movimientos de medios, entre establecimientos, dependencias o de un área de responsabilidad a otra.
Evento(s) que lo genera(n)	No Aplicable.

Pre condiciones	Se debe disponer de un modelo de Solicitud de Movimiento de medios.
Marco legal	N/A
Clientes internos	No Aplicable.
Clientes externos	No Aplicable.
Entradas	Modelo de Solicitud de Movimiento de Medios Materiales.
Flujo de eventos	
Flujo básico	
1.	Emitir solicitud MMM. Cuando por un área a la cual está asignado un medios materiales se determina efectuar un movimiento, emite el modelo Solicitud de Movimiento de Medios Materiales.
2.	Verificar solicitud. Por el número de inventario, se muestra los datos del medio y verifica que esté correcta la solicitud.
3.	Asignar número consecutivo. Si la solicitud está correcta, se asigna a la misma un número consecutivo.
4.	Emitir modelo MMM. Luego se emite el modelo de Movimiento de medios, se gestiona la firma autorizada que proceda.
5.	Efectuar movimiento. Una vez recibido el modelo autorizado, se procede a comunicar al área cedente (área de responsabilidad que cede el MM), mediante el modelo original y copias, que efectúe el traslado, recogiendo en ambos las firmas del jefe del área cedente y del área receptora (área de responsabilidad que recibe el MM). El modelo original del MMM se archiva en el área cedente.
6.	Archivar original del modelo: Se archiva el modelo original de MMM.
7.	Actualizar listado CMM. Luego procede a emitir el listado actualizado Control de Medios Materiales, archivando el original y enviando la copia al área de responsabilidad.
8.	Se desarrollan los flujos paralelos 8 a Enviar copia al área cedente y 8 b Enviar copia al área receptora.
Pos-condiciones	
1.	Se creó el modelo de SMMM, modelo MMM y el listado CMM.
Salidas	
1.	Modelo SMMM.
2.	Modelo MMM.
3.	Listado CMM.
Subprocesos	
N/A	
Pos-condiciones	
N/A	
Salidas	
N/A	
Flujos paralelos	
8 a. Enviar copia al área cedente	
1.	Enviar copia al área cedente. Se entrega una copia del modelo de Movimiento de Medios Materiales al área cedente del medios.

2.	Archivar copia del listado del área. El área emisora del medio archiva la copia del listado de Control de Medios Materiales correspondiente.
3.	Concluye el proceso Movimientos de medios.
Pos-condiciones	
N/A	
Salidas	
N/A	
8 b. Enviar copia al área receptora	
1.	Enviar copia al área receptora. Se entrega una copia del modelo de Movimiento de Medios Materiales al área receptora del medio.
2.	Archivar copia del listado del área. El área emisora del medio archiva la copia del listado de Control de Medios Materiales correspondiente.
3.	Concluye el proceso Movimientos de medios.
Pos-condiciones	
N/A	
Salidas	
N/A	
Flujos alternos	
2 a. Enviar a corregir	
1.	Enviar a corregir. Si la solicitud esta incorrecta, la regresa al área de responsabilidad que la emitió para su corrección.
2.	Corregir solicitud. El área de responsabilidad la corrige y la remite nuevamente. Volver al paso 2 del flujo básico.
Pos-condiciones	
N/A	
Salidas	
N/A	

Tabla 2: Descripción del proceso de negocio Movimientos de medios.

2.4. Modelo conceptual

El modelo conceptual ayuda a explicar cómo están relacionados los conceptos principales de la aplicación, además de identificar cuáles son estos conceptos. A la hora de realizar un modelo conceptual las preguntas de orden son: *¿Conceptos relevantes?, ¿Cuáles son?, ¿Cómo se relacionan?*

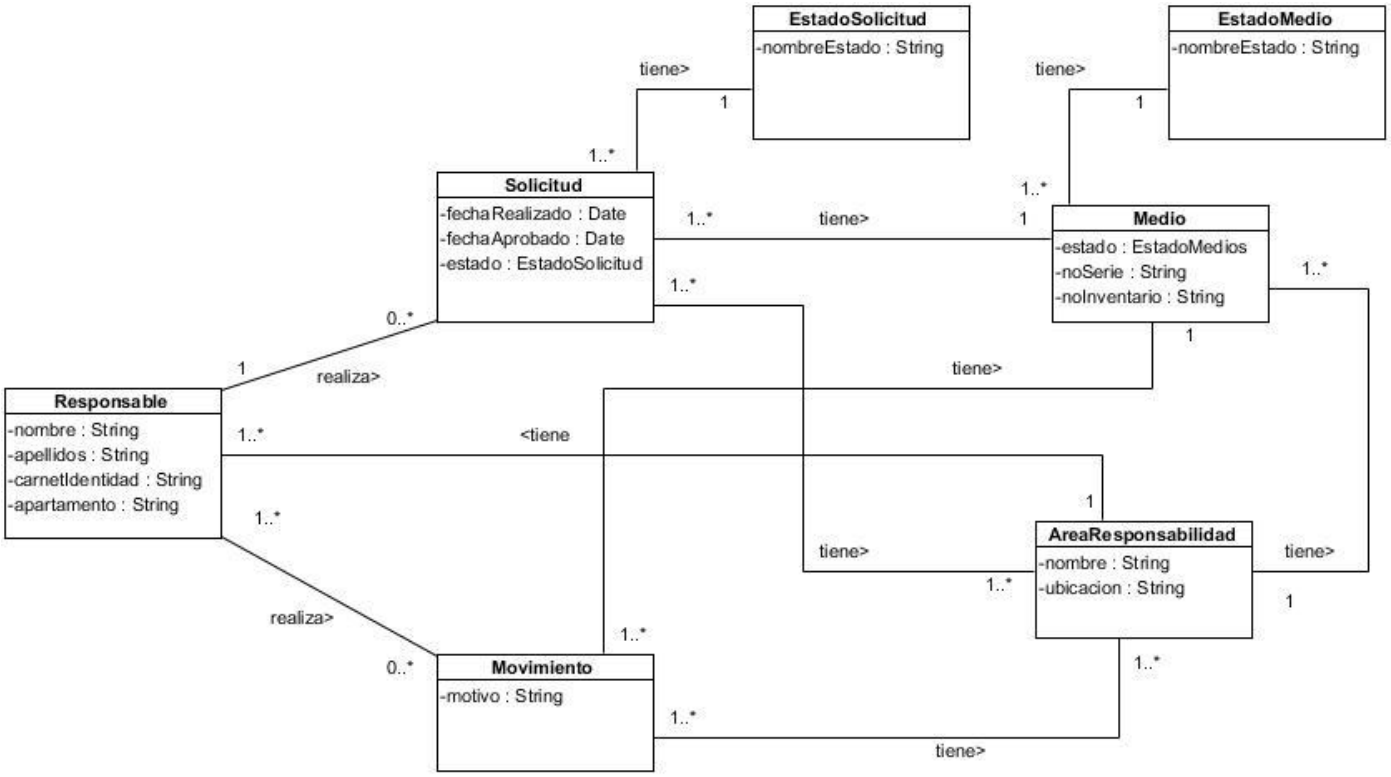

Figura 1: Modelo conceptual.

Medio: es el bien tangibles asignado al centro para su explotación, ejemplo mesas, sillas, mouse, teclados, etc.

Área de responsabilidad: local, espacio o perímetro de trabajo, ejemplo laboratorios, departamentos, oficinas, etc.

Responsable: es la persona que puede tener asignados medios en un área determinada.

Movimiento: es la acción que se realiza al mover un medio de un área de responsabilidad a otra.

Solicitud: es la petición de un medio por un responsable.

Estado de la solicitud: el estado que puede tener una solicitud, ejemplo rechazado, aprobado, etc.

Estado del medio: el estado que puede tener un medio, ejemplo roto, ocioso, etc.

2.5. Identificación de los requisitos del software

La identificación y obtención de los requisitos del software fue realizada mediante las técnicas de entrevistas, tormentas de ideas, análisis de soluciones existentes y el estudio de la documentación. Estas

no son las únicas técnicas que existen para la identificación de requisitos, sin embargo fueron las necesarias para la captura de los requisitos que se proponen incluir a la solución.

- **Entrevistas**: Las entrevistas permitieron tener un mayor conocimiento del problema y comprender los objetivos de la solución.
- **Tormenta de ideas**: Esta técnica se realizó en los primeros encuentros con el cliente, mediante una pequeña reunión que permitió obtener una visión general de las necesidades del sistema.
- **Análisis de soluciones existentes**: Mediante esta técnica se estudiaron distintos sistemas ya desarrollados que tenían características similares al sistema a desarrollar. Se analizaron los procesos en los que se involucraban los medios, dando resultados útiles para tener en cuenta en la solución.

Para lograr el desarrollo del sistema de forma satisfactoria, se partirá del análisis realizado, en el cual se identificaron los siguientes requisitos funcionales**.**

RF 01 Gestionar responsable
01.1. Registrar responsable
01.2. Actualizar responsable
01.3. Listar responsable
01.4. Buscar responsable
RF 02 Gestionar área de responsabilidad
02.1. Registrar área de responsabilidad
02.2. Actualizar área de responsabilidad
02.3. Listar área de responsabilidad
02.4. Buscar área de responsabilidad
RF 03 Gestionar medio
03.1. Registrar medio
03.2. Actualizar medio
03.3. Listar medio
03.4. Buscar medio
RF 04 Gestionar movimiento
04.1. Registrar movimiento
04.2. Listar movimiento

04.3. Buscar movimiento
RF 05 Generar reporte
RF 06 Gestionar solicitud de movimiento
06.1. Registrar solicitud de movimiento
06.2. Atender solicitud de movimiento
06.3. Listar solicitud de movimiento
06.4. Buscar solicitud de movimiento
RF 07 Gestionar solicitud de medio
07.1. Registrar solicitud de medio
07.2. Actualizar solicitud de medio
07.3. Atender solicitud de medio
07.4. Listar solicitud de medio
07.5. Buscar solicitud de medio
RF 08 Gestionar estado de medio
08.1. Registrar estado de medio
08.2. Actualizar estado de medio
08.3. Eliminar estado de medio
08.4. Listar estado de medio
08.5. Buscar estado de medio
RF 09 Gestionar estado de solicitud
09.1. Registrar estado de solicitud
09.2. Actualizar estado de solicitud
09.3. Eliminar estado de solicitud
09.4. Listar estado de solicitud
09.5. Buscar estado de solicitud
RF 10 Gestionar tipo de medio
10.1. Registrar tipo de medio
10.2. Actualizar tipo de medio
10.3. Eliminar tipo de medio
10.4. Listar tipo de medio

10.5. Buscar tipo de medio
RF 11 Gestionar tipo persona
11.1. Registrar tipo de persona
11.2. Actualizar tipo de persona
11.3. Eliminar tipo de persona
11.4. Listar tipo de persona
11.5. Buscar tipo de persona
RF 12 Gestionar ubicación
12.1. Registrar ubicación
12.2. Actualizar ubicación
12.3. Eliminar ubicación
12.4. Listar ubicación
12.5. Buscar ubicación
RF 13 Gestionar categoría
13.1. Registrar categoría
13.2. Actualizar categoría
13.3. Eliminar categoría
13.4. Listar categoría
13.5. Buscar categoría
RF 14 Realizar traslado a alta
RF 15 Realizar traslado a baja
RF 16 Realizar traslado a ocioso

2.6. Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable.

Rendimiento

- El sistema proporciona tiempos de respuesta entre 2 y 8 segundos para los procesos en línea.
- El sistema garantiza una velocidad estable de navegación a nivel de aplicación.

Seguridad

- Confiabilidad: la información manejada por el sistema está protegida de acceso no autorizado.
- Integridad: la información manejada por el sistema es objeto de cuidadosa protección contra la corrupción y estados de inconsistencia.
- Disponibilidad: la aplicación estará disponible en todo momento para aquellas personas con el acceso a la información y los mecanismos utilizados para la seguridad, no son un obstáculo para que los usuarios obtengan los datos requeridos.

Fiabilidad

- Garantizar capacidad para capturar excepciones.

Capacidad

- El sistema soporta múltiples conexiones simultáneamente.
- El sistema está previsto para el crecimiento en el volumen de datos.

Funcionalidad

- Las búsquedas en el sistema tienen un tiempo de respuesta menor de 5 segundos. Se usa la mínima cantidad de páginas para ejecutar todas las funciones posibles, es decir, se agrupan funciones afines en las mismas páginas.

Portabilidad

- El sistema es multiplataforma, es decir, puede ser utilizado tanto en los Sistemas Operativos (SO) Linux como Windows.

Software

- El servidor de procesamiento deberá contar con el framework Symfony2, así como servidor de aplicaciones web dependiendo del SO instalado.
- Las computadoras clientes deberán contar con navegadores web como Google Chrome versión 28 o inferior o Mozilla Firefox versión 21 o inferior.

- El servidor de base de datos deberá tener instalado el gestor de base de datos PostgreSQL 9.1.

Hardware

- Las computadoras clientes como los servidores deberá tener una tarjeta de red con velocidad de trasmisión mínima de 100Mbps.
- El servidor de procesamiento requiere memoria RAM de 1 GB DDR2 o superior, procesador de 3.0 GHZ o superior y capacidad de almacenamiento de 80 GB o superior.
- Las computadoras clientes requieren memoria RAM de 1 GB DDR2 o superior, procesador de 3.0 GHZ.
- El servidor de base de datos requiere memoria RAM de 1 GB DDR2 o superior, procesador de 3.0 GHZ o superior y capacidad de almacenamiento de 80 GB o superior.

2.7. Patrones arquitectónicos y de diseño

Si bien la elaboración de un buen diseño del software contribuye directamente a la calidad del producto final, gran parte de esta yace en la utilización adecuada de los patrones de diseño y arquitectónicos existentes en la actualidad. Los patrones son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos.

Con el empleo del framework Symfony2 se garantiza la utilización de varios de estos patrones, permitiendo a los desarrolladores utilizar buenas prácticas de programación y ahorrar tiempo y recursos. A continuación se describen los principales patrones empleados en el diseño e implementación de la solución.

Patrón Modelo Vista Controlador (MVC)

El patrón MVC permite dividir y organizar el código de acuerdo a las convenciones establecidas por el framework, las interfaces de usuario son manejadas en la vista, de la manipulación de datos se encarga el modelo y el procesamiento de las peticiones se manipulan en el controlador. El empleo del patrón MVC en un sistema resulta bastante útil, además de restrictivo. Symfony está basado en el patrón de arquitectura conocido MVC, formado por tres niveles: el Modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio, la Vista es la encargada de originar las páginas que son mostradas como resultado de las acciones y el Controlador se encarga de procesar las interacciones del

usuario y realiza los cambios apropiados en el modelo o en la vista. Symfony toma lo mejor de la arquitectura MVC y la implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo. La implementación que realiza Symfony de la arquitectura MVC incluye varias clases como son (Symfony, 2013):

- **Controller:** es la clase del controlador y se encarga de decodificar la petición y transferirla a la acción correspondiente.
- **Request:** guarda todos los elementos que integran la petición (parámetros, cookies, cabeceras, entre otros).
- **Response:** posee las cabeceras de la respuesta y los contenidos. El contenido de este objeto se convierte en la respuesta HTML que se remite al usuario.

Patrones GRASP

- **Controlador:** Symfony implementa un controlador frontal para atender todas las peticiones web, es el único punto de entrada de toda la aplicación en un determinado entorno. Cuando recibe una petición, utiliza el sistema de enrutamiento para enviar la acción al controlador responsable de la solución y se encarga de manejar la seguridad y ejecución de los filtros. Este patrón se evidencia en las clases Controllers y el App.php del ambiente (Symfony, 2013).
- **Alta Cohesión:** Symfony permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Un ejemplo de ello son las clases Controllers, las cuales son responsables de definir las acciones para las plantillas y colaborar con otras para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades (Symfony, 2013).
- **Bajo Acoplamiento:** las clases relacionadas con el controlador frontal y que se encuentran en el paquete Controllers heredan únicamente de Controller para alcanzar un bajo acoplamiento entre clases. Las clases que implementan la lógica del negocio y de acceso a datos se encuentran en el modelo, las cuáles no se asocian directamente con las clases de la vista o el controlador, lo que proporciona que la dependencia en este caso sea baja. Los formularios en Symfony2 solo usan los campos de las entidades que modifican, estos formularios de las páginas interna de la aplicación

son sencillos de crear ya que su única dependencia es con las entidades esto se ve evidenciado en los Type (Symfony, 2013).

- **Experto**: Symfony utiliza Doctrine2 para realizar su capa de abstracción en el modelo, encapsular toda la lógica de los datos y generar las clases con todas las funcionalidades comunes de las entidades, las clases de abstracción de datos poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla que representan (Symfony, 2013).

Patrones GoF

- **Decorator**: la clase abstracta View, padre de todas las vistas, contienen un decorador para permitir agregar funcionalidades dinámicamente. El archivo base.html.tiwg, conocido como plantilla global, contiene el código que es tradicional en todas las páginas del sistema, para no tener que repetirlo. El contenido de la plantilla se integra en el frontend, o si se mira desde otro punto de vista, el frontend decora la plantilla (Symfony, 2013).

2.8. Estándares de codificación

Un código escrito con buen estilo, en cualquier lenguaje es aquel que cumple con las siguientes propiedades:

- Está organizado.
- Es sencillo de entender.
- Es fácil de mantener.
- Posibilita la detección de errores con facilidad.

Indentación: La indentación debe ser de cuatro espacios sin usar el tabulador, así lo define symfony por defecto para clases como routing.yml y DefaultController.php, para esta última no es relevante si es tabulador o espacios en blanco.

```
activo:
    resource: "@ActivoBundle/Controller/DefaultController.php"
    type:     annotation
    prefix:   /
```

```
public function mioAction() {
    return $this->render('ActivoBundle:Default:prueba.html.twig');

}
```

Comentarios: Permite describir o deshabilitar tanto líneas, funciones, como trozos de código. Los comentarios deben ser simples, descriptivos y con buena ortografía. Los mismos pueden ser (/* */, //, {# #} o #).

```
#activo:
#    resource: "@ActivoBundle/Resources/config/routing.yml"
#    type:     annotation
#    prefix:   /
```

Estructuras de control: En cualquiera de los casos las instrucciones que pertenecen a la estructura deben estar entre "{ }" precedidas por "( )" en las cuales se indican las condiciones de las estructuras, de las cuales se incluyen: if, foreach, etc.

```
    foreach ($errores as $error) {
        $arrayError .= $error->getMessage();
    }
```

Definiciones de función: Comienza con la visibilidad, seguido de la palabra reservada "function" y el nombre de la función, la llave que abre "{" sucede a los "( )". Los parámetros van separados por coma y sin espacio del paréntesis inicial y final.

```
public function getActivosAction() {
    return $this->render('ActivoBundle:Default:activo.html.twig');
}
```

Convención de los nombres de variables, funciones y clases.

Variables: Comenzarán con letra minúscula y sin espacio, en caso de ser compuestas las demás palabras serán en mayúscula y con el prefijo "$", siguiendo la notación Camel en su variante dromedaryCase.

```
$repository = $this->getDoctrine()->getRepository('ActivoBundle:Estado');
$estados = $repository->findAll();
$response = array();
```

Funciones: Comenzarán con letra minúscula y sin espacio, en caso de ser compuestas las demás palabras serán en mayúscula y con el sufijo “Action”, siguiendo la notación Camel en su variante dromedaryCase.

```
public function getTiposActivosAction
```

Clases: Comenzarán con letra mayúscula y sin espacio, en caso de ser compuestas las demás palabras serán en mayúscula, siguiendo la notación Camel en su variante CamelCase.

```
class DefaultController
```

2.9. Historia de Usuario

La Historia de Usuario (HU) se escriben desde la perspectiva del cliente aunque los desarrolladores pueden brindar también ayuda en su descripción. La HU es el resultado de la conversación, discusión y comunicación entre el cliente y el equipo de trabajo. Su contenido debe ser concreto y sencillo (XP, 2003).

A continuación se muestra la plantilla de una HU.

Historia de usuario	
Número: 11	**Usuario:** Administrador
Nombre historia: Registrar medio.	
Prioridad en negocio: Alto	**Riesgo en desarrollo:** Alto
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Miguel Ferrer Oduardo	
Descripción: El usuario podrá agregar los medios que estarán en la aplicación.	

Observaciones:
Prototipo de interface:

Tabla 3: Representación de la historia de usuario "Registrar medio".

Las restantes historias de usuarios identificadas se pueden ver en los **Anexos** del **#1** al **#11**.

2.10. Fase planificación de la entrega

Durante la fase de planificación se realiza una estimación del esfuerzo que costará implementar cada HU. Se expresa utilizando como medida el punto. Un punto se considera como una semana (6 días) ideal de trabajo donde los miembros de los equipos de desarrollo trabajan el tiempo planeado sin ningún tipo de interrupción. Por regla general, se definió que una HU necesitan de una a tres semanas de desarrollo. Esta estimación es realizada por los programadores, que son los que están estrechamente vinculados con el desarrollo de la HU (XP, 2003).

Estimación de esfuerzo por historia de usuario

Las estimaciones proporcionarán una medida de la velocidad con la que avanza el proyecto y brinda una guía por la cual se puede ajustar el tiempo. A continuación se muestran los resultados esperados.

Historias de usuarios	Puntos de estimación (semanas)
Registrar medio	2
Gestionar movimiento	2
Gestionar solicitud de medio	3
Gestionar área de responsabilidad	1
Gestionar usuario	1
Generar reporte	2
Gestionar responsable	1
Realizar traslado a alta	1
Realizar traslado a baja	1
Realizar traslado a defectuoso	1
Realizar traslado a ocioso	1

Tabla 4: Estimación de esfuerzo por historia de usuario.

Plan de iteraciones

Una vez estimado el esfuerzo propuesto, se procede a la planificación de la etapa de implementación del sistema. Este plan define las HU que serán implementadas en cada iteración y las posibles fechas de sus entregas. En el transcurso de la fase de planificación se identificaron dos iteraciones, las cuales se especifican a continuación:

Iteración 1: Se implementan las HU que tienen mayor prioridad en el negocio, de esta forma, se van creando las funcionalidades principales del sistema que dan soporte a la implementación de las demás funcionalidades. Estas HU son: 1, 3, 10, 11, las cuales hacen alusión a los datos generales que se manejan en la aplicación, como son: gestionar usuario, gestionar área de responsabilidad, gestionar responsable, registrar medio. Además, se tendrá la primera versión de prueba, la cual será mostrada al cliente con el objetivo de obtener una retroalimentación.

Iteración 2: Se implementan las restantes funcionalidades. Con la culminación de la misma se tendrán implementadas las peticiones del cliente descritas en las HU 2, 4, 5, 6, 7, 8, 9 en las cuales se hace referencia a funcionalidades como: gestionar movimiento, gestionar solicitud de medio, generar reporte, realizar traslado a alta, realizar traslado a baja, realizar traslado a defectuoso, realizar traslado a ocioso, gestionar responsable. Se tendrá una versión de prueba, la cual al igual que la anterior se le mostrará al cliente para nuevas recomendaciones.

Plan de duración de las iteraciones

En este apéndice se mostrará la duración de las iteraciones definidas así como el orden en que se implementarán las HU.

Iteraciones	Historias de usuarios	Duración total de las iteraciones

Iteración 1	Gestionar área de responsabilidad Registrar medio Gestionar responsable	5 semanas
Iteración 2	Gestionar estado medio Gestionar estado solicitud Gestionar tipo medio Gestionar ubicación Gestionar tipo persona Gestionar categoría Gestionar movimiento Gestionar solicitud de medio Realizar traslado a alta Realizar traslado a baja Realizar traslado a ocioso Generar reporte	11 semanas

Tabla 5: Plan de duración de las iteraciones.

Plan de entrega

El propósito de este plan es agrupar las HU para conformar una entrega, las funcionalidades se reúnen referentes al mismo módulo, lo cual permite un mejor entendimiento de la implementación.

Módulos	Historias de usuarios
Usuarios	1. Gestionar responsable

Activos	1. Gestionar área de responsabilidad 2. Registrar medio 3. Gestionar movimiento 4. Gestionar solicitud de medio 5. Realizar traslado a alta 6. Realizar traslado a baja 7. Realizar traslado a defectuoso 8. Realizar traslado a ocioso 9. Generar reporte 10. Gestionar estado medio 11. Gestionar estado solicitud 12. Gestionar tipo medio 13. Gestionar ubicación 14. Gestionar tipo persona 15. Gestionar categoría

Tabla 6: Distribución de historia de usuario por módulo.

Módulos	**Final iteración 1** **3ra semana de abril**	**Final iteración 2** **3ra semana de mayo**
Usuarios	V1.0	Finalizado
Activos	V1.0	Finalizado

Tabla 7: Plan de entrega.

2.11. Diseño del sistema

Tarjetas Clases, Responsabilidad y Colaboración (CRC)

Las tarjetas CRC son una herramienta de ayuda al refinamiento de clases. El nombre de la clase se coloca en forma de título en la tarjeta, en la parte izquierda las funcionalidades (responsabilidades) y en la parte derecha las clases que se implican en cada funcionalidad (colaboración). Consiste en elaborar para cada clase una tarjeta con los siguientes datos.

- Nombre (Clase)

- Responsabilidades
- Colaboraciones

Activo	
Responsabilidades	**Colaboraciones**
Gestionar medio	
Gestionar solicitud de medio	
Gestionar movimiento	
Gestionar estado medio	MEDIO
Gestionar estado solicitud	TIPO_MEDIO
Gestionar tipo medio	ESTADO
Gestionar ubicación	TIPO_ESTADO
Gestionar tipo persona	AREA_DE_RESPONSABILIDAD
Gestionar categoría	PERSONA
Realizar traslado a alta	PERSONA_AREA_DE_RESPONSABILIDAD
Realizar traslado a baja	
Realizar traslado a ocioso	
Realizar traslado a defectuoso	

Tabla 8: Tarjeta CRC Clase Activo.

Responsable	
Responsabilidades	**Colaboraciones**
Gestionar responsable	PERSONA TIPO_DE_PERSONA NIVEL

Tabla 9: Tarjeta CRC Clase Responsable.

Area_de_responsabilidad	
Responsabilidades	**Colaboraciones**
Gestionar área de responsabilidad	AREA_DE_RESPONSABILIDAD UBICACION

Tabla 10: Tarjeta CRC Clase Área_de_responsabilidad.

2.12. Diagrama de clases del diseño

El diagrama de clases del diseño es la representación de clases y asociaciones. Los tipos de clases que se utilicen describirán la relación que tendrán al momento de la ejecución de la aplicación y como será su comunicación entre estas.

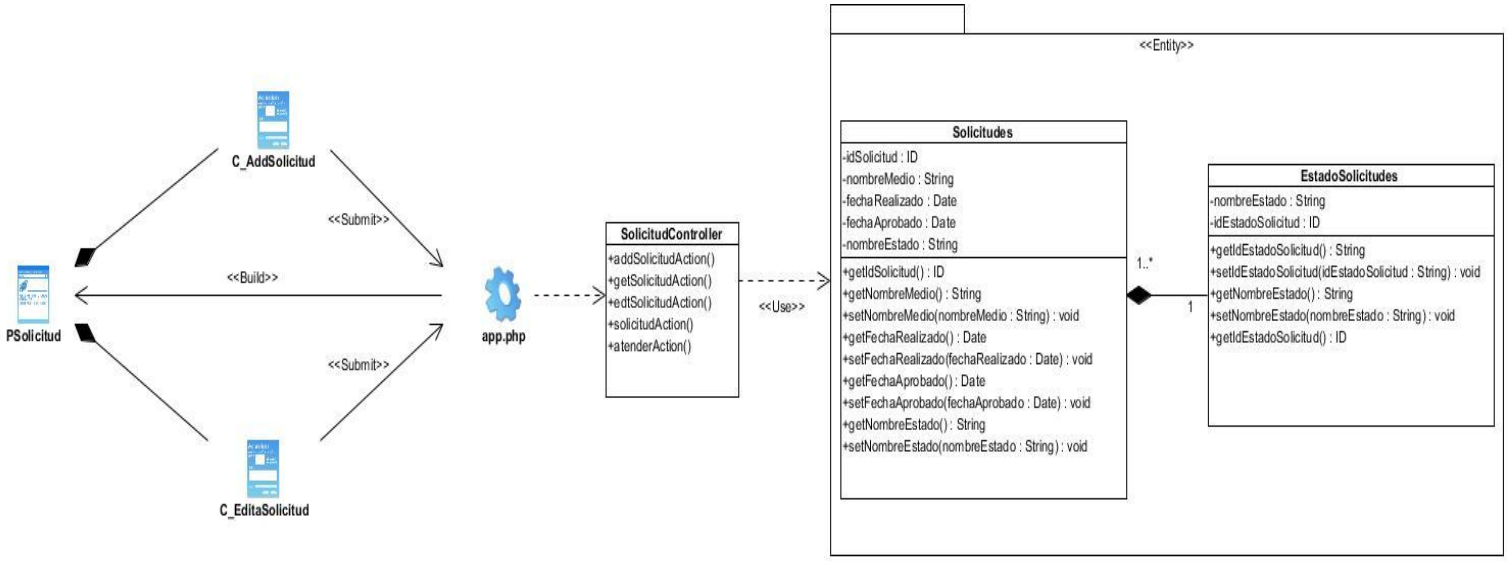

Figura 2: Diagrama de clases del diseño.

2.13. Diagrama de secuencia

Muestra como un grupo de objetos interactúan a lo largo del tiempo, dejando mensajes entre ellos y su línea de vida. El proceso de adicionar medios es uno de los más importantes en el sistema, por eso se escoge para su representación.

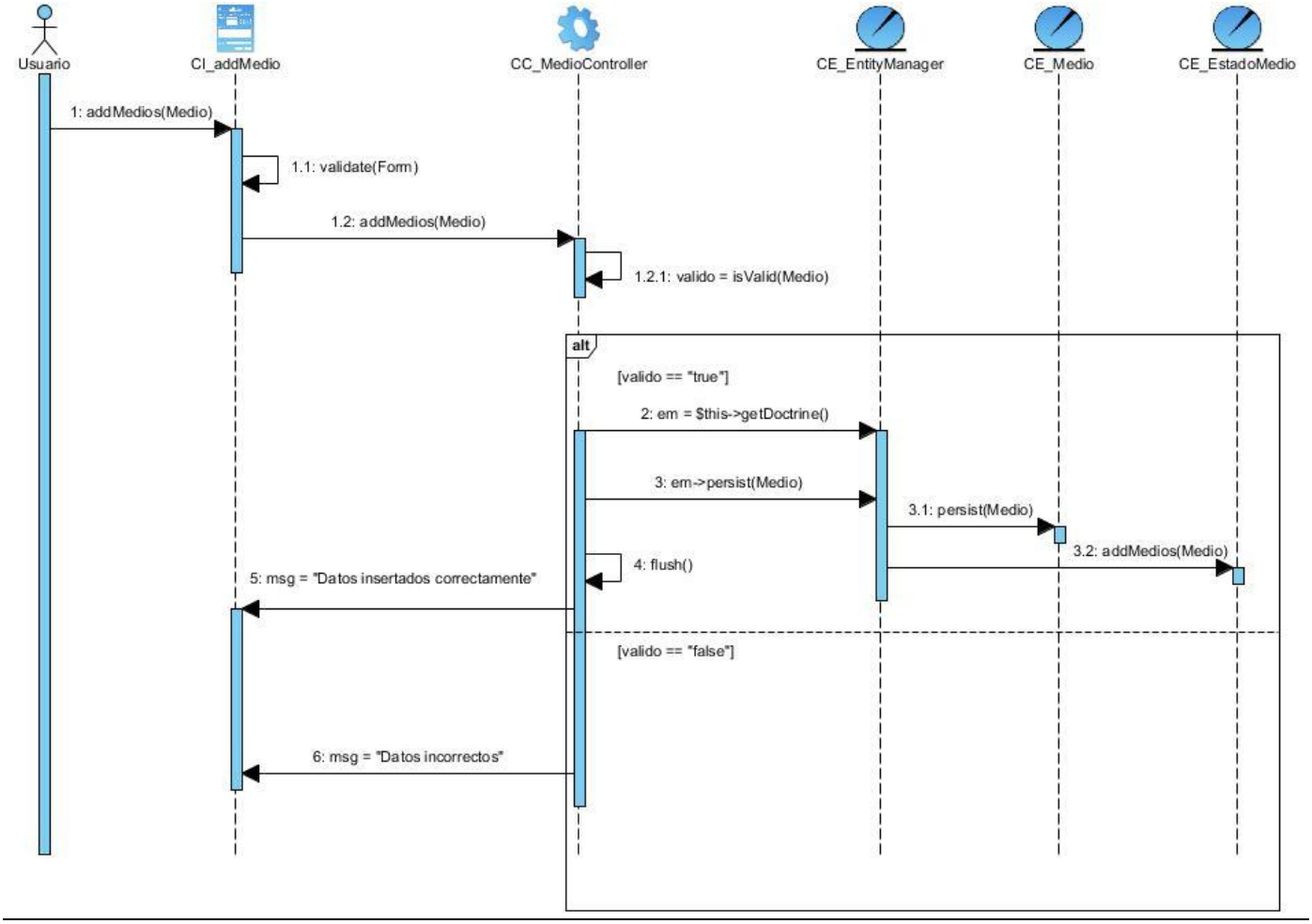

Figura 3: Diagrama de secuencia.

2.14. Diseño de la Base de Datos

El diseño de la Base de Datos (BD) es una de las tareas con mayor impacto en el desarrollo de una aplicación, con dicha representación de los datos y la relación entre los mismos ayuda a un mejor entendimiento de la solución. En otras palabras expresa la generalidad de los datos que se manejan en un sistema, agrupados en clases y la conexión entre estas últimas logrando así que funcionen como un todo.

Modelo Entidad-Relación

El modelo entidad-relación permite esquematizar la estructura de datos de forma que se pueda entender mejor lo que se está diseñando. En el modelo de la aplicación se muestran las entidades con los atributos y las relaciones entre ellas.

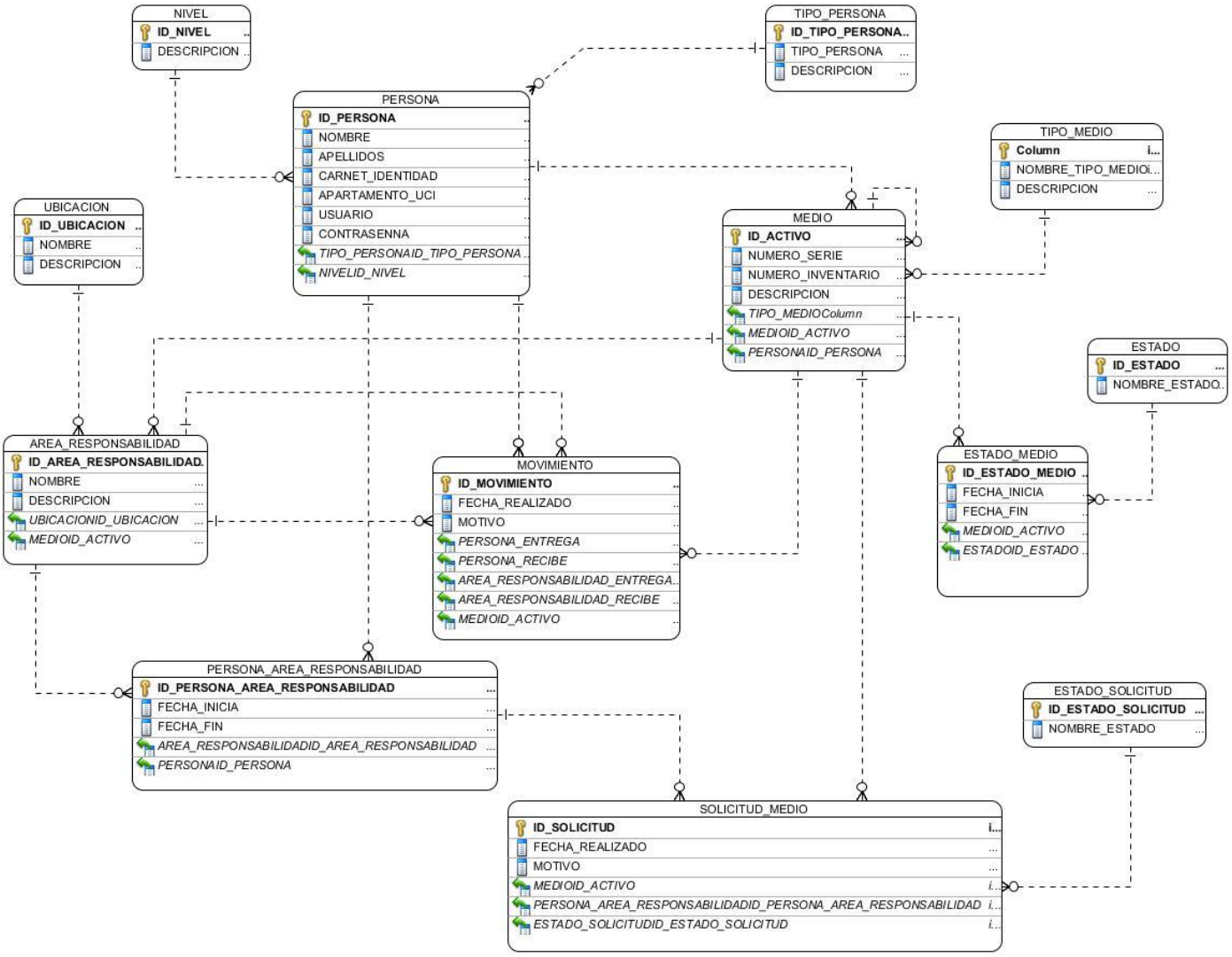

Figura 4: Modelo entidad-relación.

2.15. Diagrama de componentes

El diagrama de componentes permite con su estructura un análisis completo en la etapa de desarrollo, expresando las relaciones y dependencias de un componente. Se refleja la relación y dependencia de los

componentes del modelo, representando aspectos físicos del sistema. Se entiende como componentes a una clase de uso específico, estas pueden estar implementadas en código fuente, binario o ejecutable en el entorno de desarrollo; los tipos de componente indican si pueden ser utilizados en tiempo de compilación, enlace o ejecución (IEEE, 2010).

Para un mejor entendimiento se hará una explicación del diagrama de componentes. El diagrama representa la relación que existe entre los componentes de la aplicación y los de Symfony, el componente **Controller** usa el componente **Doctrine2** para el acceso al modelo, el **Routing** genera las notaciones para el acceso por la url, el **FOSUserBundle** es el encargado de la seguridad del sistema, también hace uso de los componente **Entity** y **Views**, el primero guarda las tablas de la BD en forma de clases y el segundo contiene las vistas que se le mostrarán al usuario y el **Form** tiene la estructura de los formularios, dicha estructura es dependiente de los atributos que se cargan de la **Entity**. El componente **Entity** usa el **Doctrine 2** para generar los métodos en cada entidad. El **Views** hace uso de **jQuery** y **Twig** para crear las plantillas. Y el **Form** usa el componente **Forms.**

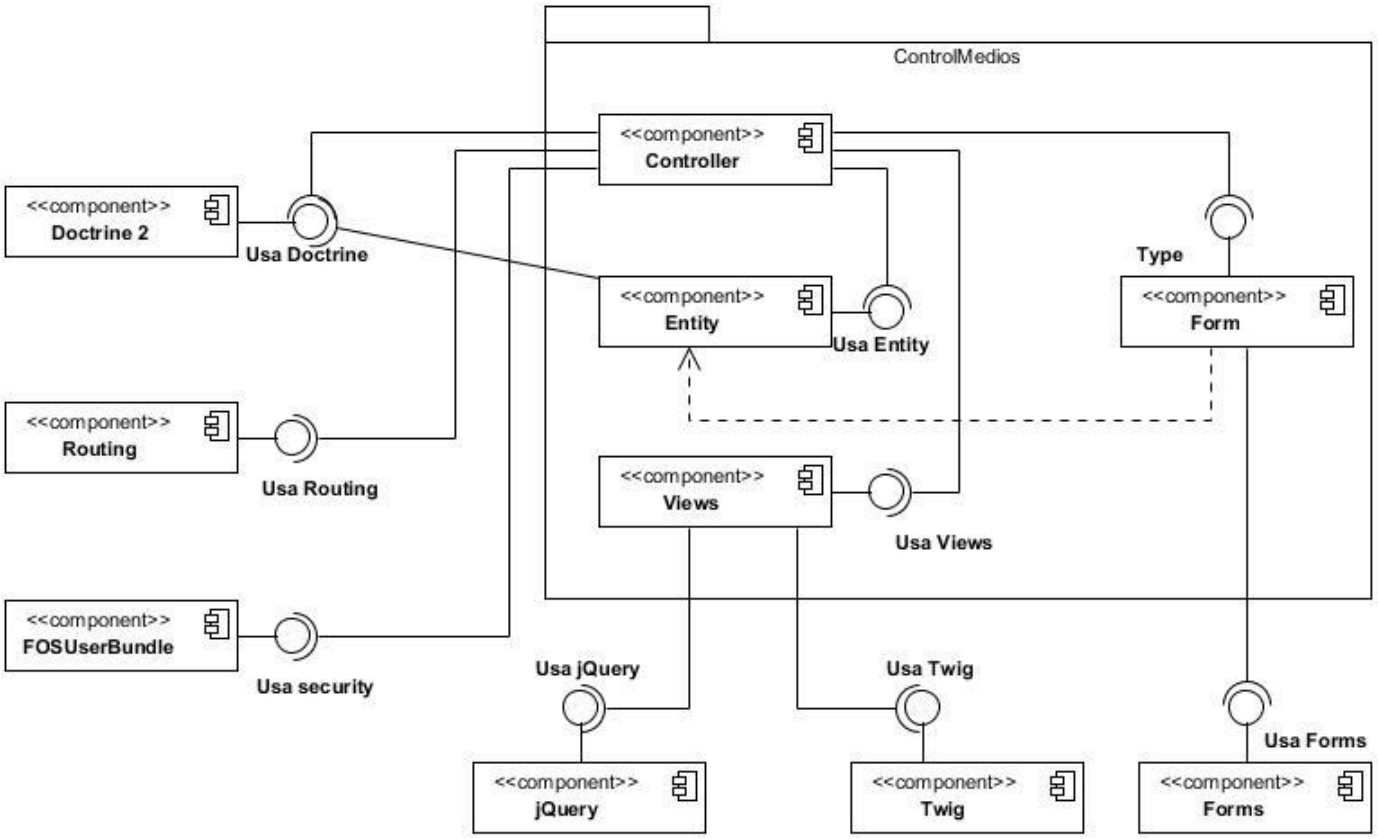

Figura 5: Diagrama de componentes.

2.16. Implementación

En este acápite se implementan todas las HU en la iteración que le corresponde, para ello sirve de apoyo el plan de iteraciones y el plan de duración de iteraciones, los cuales se revisan por si es necesario realizar cambios en ellos. Se ofrece detalles de las iteraciones desarrolladas en el sistema y las tareas que se generan en cada HU.

Tareas de programación por iteraciones

Cada HU como funcionalidad de la aplicación está compuesta por una o varias tareas de programación, estas son los pasos lógicos a seguir por el programador para realizar la implementación de una HU. A continuación se detallan para cada una de las iteraciones las tareas a desarrollar por HU.

Iteración 1

Módulos	Historias de usuario	Tiempo de Implementación (semana)	
		Estimación	Real
Usuarios	Gestionar usuario	1	1
	Gestionar responsable	1	1

Tabla 11: Tiempo real de implementación por módulo (Iteración 1).

Historias de usuario	Tareas de programación por historia de usuario
Gestionar usuarios	Registrar usuario Actualizar usuario
Gestionar área de responsabilidad	Registra área de responsabilidad Actualizar área de responsabilidad
Gestionar responsable	Registrar responsable Actualizar responsable

Tabla 12: Tareas de programación por historia de usuario (Iteración 1).

Iteración 2

Módulos	Historias de usuario	Tiempo de Implementación (semana)	
		Estimación	Real
Medio	Registrar medio	2	1.5
	Gestionar solicitud de medio	3	2.5
	Gestionar movimiento	2	1.5

Tabla 13: Tiempo real de implementación por módulo (Iteración 2).

Historias de usuario	Tareas de programación por historia de usuario
Registrar medio	Registrar medio Mostrar medio Actualizar medio

Gestionar movimiento	Registrar movimiento Atender movimiento
Generar reporte	Mostrar medios Mostrar solicitudes Mostrar movimientos
Gestionar solicitud de medio	Realizar solicitud Eliminar solicitud Actualizar solicitud
Realizar traslado a alta	Cambiar estado
Realizar traslado a baja	Cambiar estado
Realizar traslado a defectuoso	Cambiar estado
Realizar traslado a ocioso	Cambiar estado

Tabla 14: Tareas de programación por historia de usuario (Iteración 2).

2.17. Diagrama de despliegue

Permite visualizar la relación física que existe entre los componente de la aplicación, modela el hardware que se utilizó en la implementación del sistema.

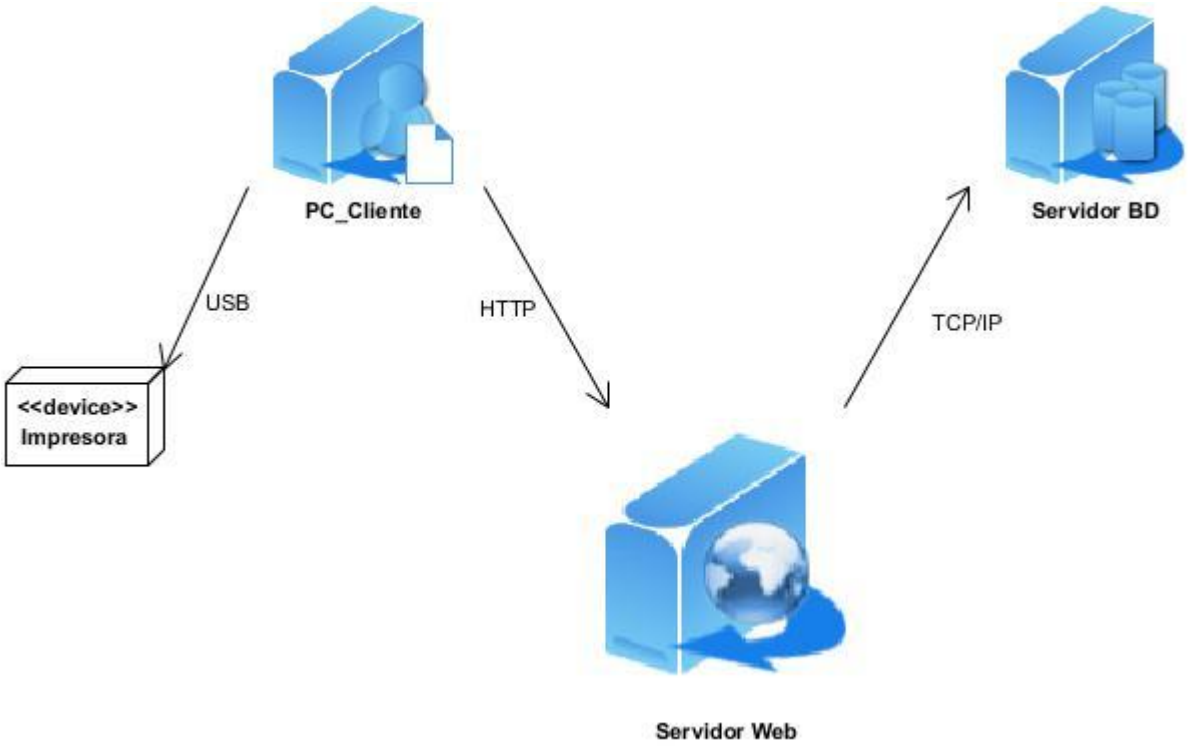

Figura 6: Diagrama de despliegue.

Descripción de los nodos

- **Cliente:** representa una computadora desde la cual el usuario podrá acceder a la aplicación.
- **Servidor Web:** representa una estación donde estará montado el servidor Apache sobre el cual se estará ejecutando la aplicación.
- **Servidor de BD:** representa el servidor donde estará el sistema gestor de base de datos Postgres SQL que dará respuesta a las peticiones hechas por la aplicación.
- **Impresora:** representa el dispositivo que imprimirá los reportes ofrecidos por la aplicación.

Protocolos

- **TCP/IP (Protocolo de control de transferencia):** se utiliza en la comunicación entre el servidor y la base de datos para realizar operaciones sobre la información de las tablas.
- **HTTP (Protocolo de transferencia de hipertexto):** establece un esquema de comunicación cliente/servidor. El cliente es el navegador web que realiza las peticiones a las que el servidor se encarga de dar respuesta.

- **USB (Bus Universal en Serie):** permite conectar periféricos para atender peticiones, además de proveer corriente si es necesario. El cliente envía una petición para imprimir un reporte.

2.18. Aspectos fundamentales de la implementación

El uso del framework Symfony2 ayuda a mantener organizado el código y permite ahorrar conexiones y dependencias innecesarias, con la ayuda del sistema de carpetas que ofrece, para el programador saber dónde está el código es una tarea bastante fácil y sencilla. La clase que contiene las funcionalidades del sistema se encuentra en la carpeta **Controller**, las plantillas y los modelos en la carpeta **views**, las clases **js** en la carpeta **gestionAFT**, las entidades y los formularios en las carpetas **Entity y Form**, respectivamente.

Figura 7: Árbol de carpetas.

La clase solicitudController.php contiene las funcionalidades, **solicitudesAction(), addSolicitudesAction()**, (ver **Anexos #12 y Anexos #13**). En la primera función el objetivo es mostrar la

vista, en la misma se mostrarán todas las solicitudes existentes en la base de datos, en cuanto a la segunda función, es la que se encarga de adicionar una nueva solicitud.

En la parte de la vista, **solicitudes.html.twig** es donde se construye la plantilla que se mostrará al usuario final, (ver **Anexos #14**), donde se muestran las opciones a escoger por el usuario del sistema, se da la opción de realizar búsquedas según diferentes criterios, (ver **Anexo #15**). Cuando se escoge una de las opciones a realizar sobre las solicitudes, se muestra el modelo **modalsAddSolicitud.html.twig** (ver **Anexo #16**), que este a su vez depende del formulario **SolicitudType.php** que caracteriza los campos, localizado en la carpeta Form (ver **Anexo #17**). En la carpeta gestionAFT el js **solicitud.js** programa el onclick de los botones dándole la funcionalidad requerida, además de realizar la validación en la vista (ver **Anexo #18**).

2.19. Conclusiones

En este capítulo se analizó en detalle la propuesta de solución que se desea implementar, además de exponer el análisis y diseño de la investigación. Se deja evidencia de cuántas funcionalidades se identificaron y en qué orden se implementaron, además de cuánto duró el desarrollo de la aplicación. Finalmente se logró la implementación del sistema orientado por los patrones, lo que permitió un trabajo organizado y un código reutilizable, así como los artefactos necesarios para un mejor entendimiento de cómo se relacionan los datos en la aplicación y que se requiere para el uso de la misma.

3. Capítulo 3: Validación de la solución propuesta

3.1. Introducción

En el presente capítulo se dará paso a la validación de la solución propuesta mediante la aplicación de las pruebas tanto al código como a la aplicación en pos de eliminar posibles errores, contribuyendo a la buena implementación y funcionamiento de los requisitos del software. Una vez terminada las pruebas se podrá definir en qué grado de funcionamiento y rendimiento se encuentra el sistema, compararlas con las exigencias del cliente y así determinar si cumplieron sus expectativas.

3.2. Pruebas

En XP el proceso de pruebas es constante, este permite controlar la calidad del sistema reduciendo el número de errores no detectados. Permite también evitar efectos no deseados a la hora de realizar cambios en el sistema.

Las pruebas se dividen en dos grupos:

1. Pruebas unitarias.
2. Pruebas de aceptación.

A petición del cliente se realizaron las pruebas de caja negra.

3.2.1. Pruebas unitarias

Las pruebas unitarias son la verificación del código y son ejecutadas por los implementadores, pues son los encargados de revisar el código directamente. El objetivo de estas pruebas es obtener un código limpio y conciso, que cumpla con cada una de las funcionalidades a las que tributan.

Como técnica se utilizó la prueba de camino básico, la cual se realizó mediante diseño de casos de prueba que se centraron en obtener una medida de la complejidad lógica del diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Estas pruebas también garantizan que en los casos de prueba obtenidos a través del camino básico, se ejecute cada sentencia del programa al menos una vez. Para esto primero se procede a enumerar las sentencias del código y a partir del mismo se construye el grafo de flujo asociado.

Para aplicar la técnica se debe introducir la notación para la representación del flujo de control, este puede representarse por un grafo de flujo en el cual:

- Cada nodo del grafo corresponde a una o más sentencias de código fuente.

- Todo segmento de código de cualquier programa se puede traducir a un grafo de flujo.
- Se calcula la complejidad ciclomática del grafo.

Un grafo de flujo está formado por 3 componentes fundamentales que ayudan a su elaboración y comprensión:

Nodos: son los círculos representados en el grafo de flujo, el cual representa una o más secuencias del procedimiento, donde un nodo corresponde a una secuencia de procesos o a una sentencia de decisión. Los nodos que no están asociados se utilizan al inicio y final del grafo.
Aristas: son constituidas por las flechas del grafo, son iguales a las representadas en un diagrama de flujo y constituyen el flujo de control del procedimiento. Las aristas terminan en un nodo, aun cuando el nodo no representa la sentencia de un procedimiento.
Regiones: son las áreas delimitadas por las aristas y nodos donde se incluye el área exterior del grafo, como una región más. Las regiones se enumeran siendo la cantidad de regiones equivalente a la cantidad de caminos independientes del conjunto básico de un procedimiento.
Para realizar la técnica planteada, específicamente la prueba del camino básico es necesario calcular antes la complejidad ciclomática del algoritmo o fragmento de código a analizar.

```
public function obtenerActivosAction() {
    $array = array(
            'success' => true
            , 'total' => 0
            , 'data' => array(
            )
        ); //1
    $activos = array(); //1
    $arregloActivos = array(); //1
    $tamannoPagina = $this->getRequest()->get('pageSize'); //1
    $paginaActual = $this->getRequest()->get('page');//1
    $elementoInicial = ($paginaActual - 1) * $tamannoPagina + 1; //1
    $text_buscar = $this->getRequest()->get('text_buscar'); //1
    $em = $this->getDoctrine()->getEntityManager(); //1
    if (is_null($text_buscar)) {            //2
        $consult = $em->createQuery('SELECT COUNT(a) FROM ActivoBundle:Activo a'); //3
        $total = $consult->getResult();//3
        $array = array(
            'success' => true
            , 'total' => $total[0][1]
            , 'data' => array(
            )
        ); ////////////////////////////////////////////////////////////////////////////3
        if (is_null($tamannoPagina) || $tamannoPagina < 1) //4
            $tamannoPagina = isset($tamannoPagina) ? $tamannoPagina : 5;  //5
```

```
    $query = $em->createQuery("SELECT DISTINCT a FROM ActivoBundle:Activo a ORDER BY a.idActivo ASC"); //6
    $activos = $query->setFirstResult($elementoInicial - 1)
    ->setMaxResults($tamannoPagina)->getResult(\Doctrine\ORM\Query::HYDRATE_OBJECT); //6
}else{
    $id_area_resp = $this->getRequest()->get('id_area_resp'); //7
    $id_estado = $this->getRequest()->get('id_estado'); //7
    $id_responsable = $this->getRequest()->get('id_responsable'); //7
    $id_tipo_medio = $this->getRequest()->get('id_tipo_medio'); //7
    $activos = $em->getRepository("ActivoBundle:Activo")
    ->findByParams($text_buscar, $id_area_resp, $id_estado, $id_responsable, $id_tipo_medio); //7
}
foreach ($activos as $activo) { //8
    $idPadre = ""; //9
    $descripPadre = ""; //9
    if ($activo->getActivoPadre() != null) { //10
        $idPadre = $activo->getActivoPadre()->getIdActivo(); //11
        $descripPadre = $activo->getActivoPadre()->getDescripcion(); //11
    }
```

```
    $arregloActivos[] = array(
        "idActivo" => $activo->getIdActivo(),
        "numeroSerie" => $activo->getNumeroSerie(),
        "numeroInventario" => $activo->getNumeroInventario(),
        "tipoActivo" => $activo->getTipoActivo()->getIdTipoActivo(),
        "tipoActivoDescrip" => $activo->getTipoActivo()->getDescripcion(),
        "descripcion" => $activo->getDescripcion(),
        "persona" => $activo->getPersona()->getIdPersona(),
        "personaDescrip" => $activo->getPersona()->getNombre() . " " . $activo->getPersona()->getApellidos(),
        "idEstado" => $activo->getEstadoActivosActual()->getEstado()->getIdEstado(),
        "estado" => $activo->getEstadoActivosActual()->getEstado()->getNombreEstado(),
        "areaResponsabilidad" => $activo->getAreaResponsabilidad()->getIdAreaResponsabilidad(),
        "areaResponsabilidadDescrip" => $activo->getAreaResponsabilidad()->getNombreArea(),
        "activoPadre" => $idPadre,
        "activoPadreDescrip" => $descripPadre
    );////////////////////////////////////////////12
}
$array['data'] = $arregloActivos; //13
return new Response(json_encode($array), 200); //14
```

Figura 8: Código del método para el camino básico.

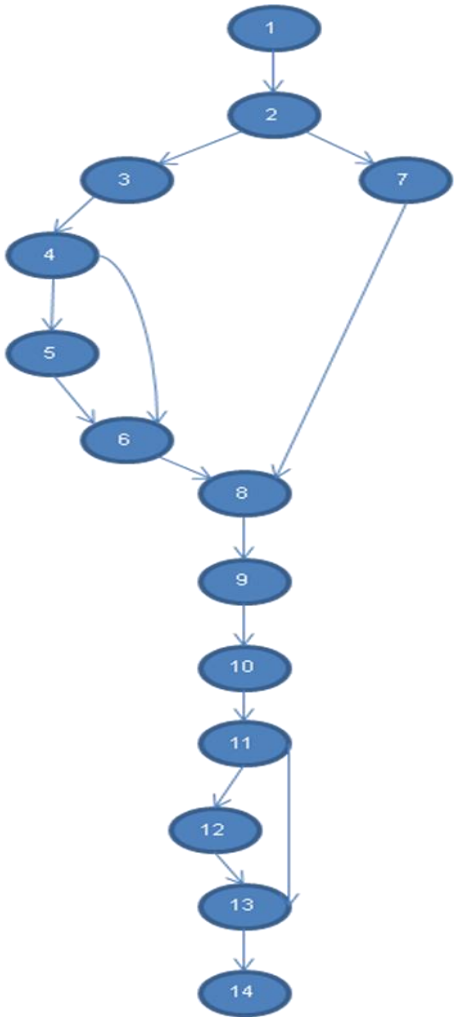

Figura 9: Grafo de flujo.

La complejidad ciclomática es extremadamente útil pues proporciona una medición cuantitativa de la complejidad lógica del código seleccionado. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. El cálculo de la complejidad se realiza mediante tres fórmulas las cuales deben arrojar el mismo resultado, lo que indica que el cálculo de la complejidad ciclomática es correcto. Dichas fórmulas se describen a continuación:

Fórmula 1 V (G) = (A - N)+2

Donde “A” es la cantidad total de aristas y “N” la cantidad total de nodos.

Fórmula 2 V (G) = P + 1

Donde “P” es la cantidad total de nodos predicados o sea de los que parten dos o más aristas.

Fórmula 3 V (G) = R

Donde “R” es la cantidad total de regiones, dígase regiones al espacio interior creado entre nodos conectados incluyendo la región exterior del grafo en su totalidad.

Fórmula 1:

= (16 – 14) + 2

= (2) + 2

= 4

Fórmula 2:

= 3 + 1

= 4

Fórmula 3:

= 4

El cálculo arrojó un valor de cuatro para las tres fórmulas, lo cual da la certeza de que existen los cuatro caminos básicos siguientes.

Camino básico #1 (1-2-3-4-5-6-8-9-10-11-12-13-14)

Camino básico #2 (1-2-3-4-6-8-9-10-11-12-13-14)

Camino básico #3 (1-2-7-8-9-10-11-12-13-14)

Camino básico #4 (1-2-7-8-9-10-12-13-14)

A continuación se realizará un caso de prueba por cada camino básico, teniéndose en cuenta las siguientes exigencias:

Descripción: se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.

Condición de ejecución: se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.

Resultados Esperados: se expone el resultado que se espera que devuelva el procedimiento.

Evaluación de los resultados: se exhibe la evaluación que dio el resultado final del procedimiento.

Caso de prueba. Camino básico #1:

Descripción: para ejecutarse no necesita parámetros, se retornarán en un grid todos los datos de los medios existentes en la base de datos, además de los criterios de búsqueda y sus resultados.

Condición de ejecución: debe existir al menos un medio en base de datos. La variable $text_buscar es nula, la variable $tamannoPaguina es nula o un entero negativo. La función getActivoPadre() es distinta de nulo.

Entrada: arreglo de medios.

Resultados esperados: un listado con todos los medios existentes en diferentes páginas de 10 en 10.

Caso de prueba. Camino básico #2:

Descripción: para ejecutarse no necesita parámetros, se retornarán en un grid todos los datos de los medios existentes en la base de datos, además de los criterios de búsqueda y sus resultados.

Condición de ejecución: la variable $text_buscar es nula. El valor de la variable $tamannoPaguina no debe ser nula, ni un entero negativo.

Entrada: arreglo de medios.

Resultados esperados: un listado con todos los medios existentes en páginas diferentes, en razón de 10.

Caso de prueba. Camino básico #3:

Descripción: para ejecutarse no necesita parámetros, se retornarán en un grid todos los datos de los medios existentes en la base de datos, además de los criterios de búsqueda y sus resultados.

Condición de ejecución: debe existir al menos un medio en base de datos. La variable $text_buscar no es nula.

Entrada: arreglo de medios. Criterios de búsqueda.

Resultados esperados: un listado en una sola página con el resultado de la búsqueda avanzada.

Caso de prueba. Camino básico #4:

Descripción: para ejecutarse no necesita parámetros, se retornarán en un grid todos los datos de los medios existentes en la base de datos, además de los criterios de búsqueda y sus resultados.

Condición de ejecución: debe existir al menos un medio en base de datos. La variable $text_buscar no es nula.

Entrada: arreglo de medios. Criterios de búsqueda.

Resultados esperados: un listado en una sola página con el resultado de la búsqueda avanzada.

Evaluación de los resultados: realizada las pruebas a la funcionalidad se obtuvieron los resultados esperados, cumpliéndose la ejecución de cada camino al menos una vez con las salidas esperadas.

3.2.2. Prueba de aceptación

Las pruebas de aceptación son diseñadas a partir de cada iteración y basadas en las HU, en ellas el cliente es el encargado de comprobar que ha sido correctamente implementada. Estas pruebas aseguran que las funcionalidades del sistema cumplan con lo que se espera de ellas, marcan el camino a seguir indicando las funcionalidades más importantes, permiten que el cliente sepa cuando el sistema está funcionando correctamente, además que los programadores conozcan qué les resta por hacer. Tienen una importancia crítica para el éxito de una iteración.

Caso de prueba de aceptación	
Código: HU2_P1	**No. HU: 2**
Nombre: Insertar un movimiento	
Descripción: Prueba de funcionalidad que permite insertar un movimiento.	
Condiciones de ejecución: El usuario debe estar autenticado como un administrador del sistema, debe existir al menos un medio y dos áreas de responsabilidad. El medio debe pertenecer a una de las áreas involucradas.	
Escenario #1: **Entrada/Pasos de ejecución:** El usuario llena los campos correctamente. **Resultados esperados:** El sistema muestra un mensaje indicando que la acción fue satisfactoria. El movimiento es insertado en la base de datos. El movimiento es mostrado en un listado.	
Escenario #2: **Entrada/Pasos de ejecución:** El usuario introduce campos incorrectos. **Resultados esperados:** El sistema muestra un mensaje indicando que existen campos	

incorrectos. El sistema permite introducir los campos nuevamente.
Escenario #3: **Entrada/Pasos de ejecución:** El usuario deja campos vacíos. **Resultados esperados:** El sistema muestra un mensaje indicando que existen campos vacíos. El sistema permite introducir los campos nuevamente.
Evaluación de prueba:

Tabla 15: Caso de prueba "Insertar movimiento".

En los **Anexo #19** al **Anexo #33** se pueden encontrar los restantes casos de prueba.

Algunas de las reuniones con el cliente se dedicaron a probar el correcto funcionamiento de los requisitos implementados, a cada uno de estos encuentros le denominaremos iteraciones.

Para una primera iteración se detectaron un conjunto de 6 no conformidades, para las cuales el cliente dio las sugerencias de cómo deberían funcionar correctamente. Para la segunda iteración habiéndose corregido las no conformidades detectadas anteriormente, se señalaron 2 no conformidades. Para una última iteración no se detectó ningún mal funcionamiento, logrando así la satisfacción del cliente y obteniendo un sistema funcional.

3.2.3. Pruebas de caja negra

Las pruebas de Caja Negra deben su nombre a los elementos que estas revisan y las condiciones en que se hace la revisión. Estas se basan en los requerimientos funcionales del sistema y se llevan a cabo desde el exterior de la aplicación. Este tipo de prueba es importante a la hora de medir el grado de cumplimiento de los requerimientos solicitados por el cliente y se aplican sobre la interfaz de la aplicación, observando las respuestas del sistema ante determinadas acciones (Pressman, 2004).

Después de realizadas las pruebas de aceptación a petición del cliente se le realizaran las pruebas de caja negra al sistema en el laboratorio de pruebas internas del centro, en las cuales no se encontraron no conformidades, como evidencia en el (**Anexo #34**) se encuentra el acta de liberación del sistema.

3.3. Conclusiones

En el desarrollo del capítulo se realizaron las pruebas de aceptación, con esta práctica se aseguró que la aplicación cumple con las expectativas del cliente, se crearon un conjunto de casos de pruebas reflejando el correcto funcionamiento del sistema las cuales sirvieron de guía a los probadores. Las pruebas unitarias se realizaron usando la técnica de camino básico, las cuales arrojaron resultados satisfactorios sobre el funcionamiento del código implementado. Para esto se calculó la complejidad ciclomática de uno de los métodos más significativos de la aplicación, asegurando que la ejecución de cada camino ocurra al menos una vez.

Conclusiones generales

La definición de conceptos asociados a la gestión de los medios materiales, ayudó a una mejor comprensión del objetivo a cumplir. La investigación y análisis de las herramientas permitió seleccionar las idóneas para el desarrollo del trabajo; así como definir la necesidad de esta nueva solución ajustada a las necesidades del CEIGE.

El uso de la metodología XP para guiar el proceso y los artefactos generados en cada una de sus fases, propició una mayor comunicación entre el equipo de desarrollo y el cliente, logrando obtener un mejor funcionamiento del sistema y que éste responda a todas las necesidades.

Con el presente trabajo se propuso el desarrollo de un sistema para gestionar los medios en el CEIGE. Como resultado del mismo se ofrecen las siguientes ventajas:

- Obtener búsquedas y exportar a PDF el resultado de las mismas.
- Humanizar el trabajo de las personas encargadas del control de los medios en el centro.
- La estandarización de los nomencladores de los medios.
- Respuestas ágiles y verídicas.
- Generar reportes de históricos, movimientos y actas de responsabilidad.

Con la realización de las pruebas al sistema se logró corregir todos los errores detectados, esto permitió satisfacer las necesidades del cliente y lograr una solución funcional. Por lo que se considera concluida la investigación y dado por cumplidos todos los objetivos establecidos al comienzo de este trabajo.

Recomendaciones

A partir de la experiencia acumulada en el desarrollo de este trabajo se propone:

- ✓ Hacer extensiva el uso de la herramienta obtenida a la facultad 3.
- ✓ Usar la solución como apoyo en la toma de decisiones.
- ✓ La publicación de la solución.

Referencias bibliográficas

[1]. ABC, 2012. Definición de recursos materiales - Qué es, Significado y Concepto. In: [online]. 16 November 2012. [Accessed 16 November 2012]. Available from: http://definicion.de/recursos-materiales/ http://www.definicionabc.com/general/recursos-materiales.php.

[2]. ANON., 1999. Anarchism Triumphant. In: [online]. 1999. [Accessed 23 November 2012]. Available from: http://emoglen.law.columbia.edu/publications/anarchism.html.

[3]. CONSEJO DE ESTADO, 2007a. *Compendio Legislativo. Responsabilidad Material* [online]. 23 July 2007. S.l.: s.n. Available from: http://correo.servisa.tur.cu/Capacitacion/Consultor/09%20Legislacion/Decreto%20Ley/DL-2007-249.htm.

[4]. CONSEJO DE ESTADO, 2007b. *Ministerio del trabajo y seguridad social.* 16 August 2007. S.l.: s.n.

[5]. FAO, 2009. *Publicaciones de Medio Ambiente, Cambio Climático y Bioenergía*. S.l.: s.n.

[6]. IEEE, 2010. Diagramas UML: Componentes y despliegue. In: [online]. 26 September 2010. [Accessed 10 May 2013]. Available from: http://www.slideshare.net/joshell/diagramas-uml-componentes-y-despliegue,.

[7]. MARTÍN, Santiago Escárez San, 2008. Inventario de Recursos y Medios. In: [online]. 2008. [Accessed 30 May 2013]. Available from: www.plandeemergencias.com www.gestiondelaseguridad.com www.eladministrador.cl.

[8]. MICROSOFT, 2012. Introducción al lenguaje de programación Visual Basic. In: [online]. 2012. [Accessed 1 June 2013]. Available from: http://msdn.microsoft.com/es-es/library/xk24xdbe%28VS.80%29.aspx.

[9]. SYMFONY, 2013. *Symfony_book 2.2* [online]. S.l.: s.n. Available from: http://github.com/symfony/symfony-docs/issues.

[10]. VOX LAROUSSE, 2013. Resultados para recursos materiales en un diccionario. In: [online]. 2013. [Accessed 28 May 2013]. Available from: http://www.diccionarios.com/detalle.php?palabra=recursos+materiales&dicc_100=on&Buscar.x=0&Buscar.y=0&palabra2=.

[11]. VTIGER, 2012. CRM | vtiger CRM Software. In: [online]. 2012. [Accessed 1 June 2013]. Available from: http://www.vtiger-deutschland.de/vtigerdocu.html.

[12]. XP, 2003. XP Agile Universe. Conference on eXtreme Programming and Agile Processes in Software Engineering. Agile Development Conference (EEUU). Agile Development Conference (Australia). In: [online]. 2003. [Accessed 1 June 2013]. Available from: www.agileuniverse.com www.xp2004.org www.agiledevelopmentconference.com www.softed.com/adc2003/.

[13]. Acosta, Romero and Medina, Porras, 2007. Openbravo. Bogotá D.C.: Universidad Nacional Colombia, Facultad De Ciencias Económicas, Unidad De Informática y Comunicaciones. In: 2007.

[14]. ANON., 1999. Anarchism Triumphant. In: [online]. 1999. [Accessed 23 November 2012]. Available from: http://emoglen.law.columbia.edu/publications/anarchism.html.

[15]. ANON., 2002. Rodas XXI. Sistema Integral Económico Administrativo. In: [online]. 2002. [Accessed 23 November 2012]. Available from: http://www.rodasxxi.cu/.

[16]. ANON., 2006. GESPRO - Entorno de Desarrollo. In: [online]. 2006. [Accessed 23 November 2012]. Available from: http://gespro.prod.uci.cu/.

[17]. APPELO, Jurgen, 2008. The Definitive List of Software Development Methodologies. In: [online]. 2008. [Accessed 29 November 2012]. Available from: http://www.noop.nl/.

[18]. ASSETS, 2013. ASSETS: Sistema de Gestión Integral. In: [online]. 2013. [Accessed 4 February 2013]. Available from: http://www.assets.co.cu/assets.asp.

[19]. CASTLEDINE, EARLE and SHARKIE, 2010. jQuery: Novic e to Ninja. Cambridge. 2010. S.l.: s.n.

[20]. CHRISTIANSEN and SIMON, 2012. Database Management System (DBMS). In: [online]. 2012. [Accessed 30 November 2012]. Available from: http://searchsqlserver.techtarget.com/.

[21]. ECLIPSE FOUNDATION, 2012. Eclipse: The Eclipse Foundation. About the Eclipse Foundation. In: [online]. 2012. [Accessed 30 November 2012]. Available from: http://www.eclipse.org/org/.

[22]. FAJARDO UGARTE, Jorge, 2008. BPMN: estandar para modelamiento de procesos. 2008. S.l.: s.n.

[23]. JACOBSON, 1998. Applying UML in The Unified Process. 1998. S.l.: s.n.

[24]. JESUS GARCIA, 2011. Ext JS in Action: Jesus Garcia: 9781935182115: Amazon.com: Books. In: [online]. 2011. [Accessed 24 January 2013]. Available from: http://www.amazon.com/Ext-JS-Action-Jesus-Garcia/dp/1935182110.

[25]. MSC.PORTEIRO and LIC.MORALES, 2008. Casa Consultora DISAIC - content. In: [online]. 2008. [Accessed 23 November 2012]. Available from: http://www.disaic.cu/modules.php?name=content&pa=showpage&pid=818.

[26]. ORACLE, 2012. JD Edwards World | Aplicaciones | Oracle. In: [online]. 2012. [Accessed 24 January 2013]. Available from: http://www.oracle.com/lad/products/applications/jd-edwards-world/index.html.

[27]. ORACLE CORPORATION, 2012. MySQL:: FOSS License Exception. In: [online]. 2012. [Accessed 30 November 2012]. Available from: http://www.mysql.com/about/legal/licensing/foss-exception/.

[28]. POTENCIER, Fabien, 2008. La Guía Definitiva de Symfony. 2008. S.l.: s.n.

[29]. PRESSMAN, Roger S, 2004. Ingenería del Software. Un enfoque práctico. Quinta Edición. s.l.: McGraw Hill,. S.l.: s.n.

[30]. RIGGS, SIMON, KROSIGN and HANNU, 2010. PostgreSQL 9 Administration Cookbook. Birmingham. 2010. S.l.: s.n.

[31]. SOFTWARE DEVELOPMENT METHODOLOGIES, 2012. Software Development Methodologies. In: [online]. 2012. [Accessed 29 November 2012]. Available from: http://softwaredevelopmentmethodologies.org/.

[32]. SONIA SÁNCHEZ ANDUJAR, Sonia Sánchez Andujar, 2008. PROCEDIMIENTO: GESTIÓN DE LOS RECURSOS MATERIALES. Facultad de Ciencias Sociales y Jurídicas.Universidad de Jaén. In: 28 April 2008.

[33]. SOUSA, Susan, 2009. My PM Expert. The Advantages and Disadvantages of Agile SCRUM Software Development. In: [online]. 2009. [Accessed 29 November 2012]. Available from: http://www.my-project-management-expert.com/.

[34]. THE JQUERY PROJECT, 2010. jQuery: The Write Less, Do More, JavaScript Library. In: [online]. 2010. [Accessed 30 November 2012]. Available from: http://jquery.com/.

[35]. THE PHP GROUP, 2012. PHP: Hypertext Preprocessor. In: [online]. 2012. [Accessed 29 November 2012]. Available from: http://www.php.net/.

[36]. VERA, VERA, 2006. Implementación De Sistemas ERP, Su impacto en la gestión de la empresa e integración con otras TIC. Universidad de Concepción: CAPIV REVIEW Vol. 4. 2006. S.l.: s.n.

[37]. VISUAL PARADIGM, 2012. UML, BPMN and Database Tool for Software Development. In: [online]. 2012. [Accessed 29 November 2012]. Available from: http://www.visual-paradigm.com/.

[38]. WESKE, Mathias, 2007. Business Process Management. Postdam: Springer, 2007. ISBN 978 -3 -540 - 73521 -2. 2007. S.l.: s.n.

[39]. ZAPATA, Carlos Mario, 2009. Scielo [online]. 2009. S.l.: s.n. Available from: http://www.scielo.org.co/pdf/ring/n29/n29a3.pdf.

[40]. ZAVALA RUIZ, 2008. Análisis organizacional de empresas de software. Una primera aproximación a una visión alterna de la Ingeniería de Software. Universidad Autónoma Metropolitana. México. 2008. S.l.: s.n.

[41]. ZEND TECHNOLOGIES LTD, 2012. Zend Framework. In: [online]. 2012. [Accessed 29 November 2012]. Available from: http://framework.zend.com/about.

Bibliografía

JACOBSON, 1998. *Applying UML in The Unified Process.* 1998. S.l.: s.n.

ZAVALA RUIZ, 2008. *Análisis organizacional de empresas de software. Una primera aproximación a una visión alterna de la Ingeniería de Software. Universidad Autónoma Metropolitana. México*. 2008. S.l.: s.n.

POTENCIER, Fabien, 2008. *La Guía Definitiva de Symfony.* 2008. S.l.: s.n.

SYMFONY, 2013. *Symfony_book 2.2* [online]. S.l.: s.n. Available from: http://github.com/symfony/symfony-docs/issues.

PRESSMAN, Roger S, 2004. *Ingenería del Software. Un enfoque práctico.* Quinta Edición. s.l. : McGraw Hill,. S.l.: s.n.

RIGGS, SIMON, KROSIGN and HANNU, 2010. *PostgreSQL 9 Administration Cookbook. Birmingham*. 2010. S.l.: s.n.

SOFTWARE DEVELOPMENT METHODOLOGIES, 2012. Software Development Methodologies. In: [online]. 2012. [Accessed 29 November 2012]. Available from: http://softwaredevelopmentmethodologies.org/.

SONIA SÁNCHEZ ANDUJAR, 2008. PROCEDIMIENTO: GESTIÓN DE LOS RECURSOS MATERIALES. Facultad de Ciencias Sociales y Jurídicas.Universidad de Jaén. In: 28 April 2008. from: http://www10.ujaen.es