Universidad de las Ciencias Informáticas
Centro de Consultoría y Desarrollo de Arquitecturas Empresariales (CDAE) , Facultad 5

Diseño e implementación de un módulo para la gestión de activos para la plataforma GESPRO 16.05.

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

Autor: Ernesto Soler Calaña

Tutores: Ing. Rosel Sosa González

Ing. Yadira Tasé Torres

La Habana, 2016

El valor de una educación universitaria no es el aprendizaje de muchos datos, sino el entrenamiento de la mente para pensar.
Albert Einstein.

Dedicatoria

A mis padres: Maritza, Enrique, Aleida, Chino y Arsenio quienes han contribuido a mi formación como persona. Gracias por la confianza, por el amor, por el apoyo, por su dedicación y educación brindada. Soy lo que soy gracias a ustedes. MIL GRACIAS!!. Los quiero mucho a todos.

En especial, a mi abuela Cuca, por estar a mi lado en todo momento e iluminarme el camino. Siempre te llevaré en mi corazón.

A mi amiga, compañera de guerras, mi niña, mi amor, mi futura esposa y madre de mis hijos: Yadira Tasé Torres, por su amor y dedicación todos estos años. Te Amo Pochi.

A mis tres hermanos Hecty, Suse y Sergi por su preocupación y amor.

A mis tíos, tías, primos y familia en general por su apoyo incondicional.

A mis dos queridos sobrinos: Diego y Enzo, por ser tan hermosos.

Agradecimientos

Le agradezco a ...

Al Dr. C Pedro Yobanis Piñero Pérez, por confiar en mi la realización de esta investigación, por su tutoría extra-oficial y sobre todo por permitirme el placer de trabajar con su equipo de proyecto. Le estaré eternamente agradecido.

A mi tribunal de los tres cortes de tesis, por sus fuertes críticas (todas contructivas) y por su profesionalidad en todo momento.

A mis dos tutores: Ing. Rosel Sosa González y la Ing. Yadira Tasé Torres, por su ayuda incondicional y apoyo en el desarrollo de la tesis.

A mis profesores de la carrera por sus excelentes clases y ayuda a lo largo de mi vida universitaria.

Al equipo de proyecto de GESPRO por su ayuda y profesionalidad, en especial : Javier, Félix, Michael y la Dr. Anisleiby.

Al profe Hassán Lombera Rodríguez por la ayuda brindada en LATEX.

A mis tías Diana y Bertica por su ayuda tan oportuna y el cariño brindado. Les estaré eternamente agradecido.

A mi cuñado William por asistir a mi tesis y por su amistad... Johnny be good!!!

A mis suegros y abuela Migdalia incluida por su preocupación y por ser tan maravillosos conmigo.

A Mabel por su apoyo y por ser tan atenta conmigo .

A mis amistades Delis y Yaris por ser tan bellas personas conmigo.

A mis amistades del IPI, en especial a Ernestón y Yordan por su amistad y los buenos momentos.

A mis compañeros de la carrera desde el grupo 5103 hasta el 5503, por compartir estos años conmigo.

A todas aquellas personas que han estado pendiente durante mis cinco años de estudios universitarios y me han ayudado a alcanzar el éxito. ¡Muchas Gracias!

Declaración de autoría

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

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

Ernesto Soler Calaña
Autor

Ing. Rosel Sosa González
Tutor

Ing. Yadira Tasé Torres
Tutora

Resumen

La presente investigación propone el desarrollo de un módulo para la gestión de activos para la plataforma GESPRO 16.05 . La solución propuesta permite suplir las actuales limitaciones del módulo de Alcance Trazabilidad de la versión 14.05 de GESPRO, tales como: gestión de activos tangibles, cálculo del valor actual y la depreciación/amortización acumulada en tiempo real para los activos registrados en el sistema.

Para el desarrollo de la investigación y cumplimiento del objetivo propuesto se utiliza la Norma ISO 55000 para la gestión de activos que provee terminologías y beneficios a las organizaciones, además se abordan conceptos relacionados con el tema de la investigación, que posibilitan asociar la definición de reutilización con la gestión de activos. Se utiliza la metodología ágil SCRUM que responde a los requisitos del cliente.

Se definen los requisitos funcionales y no funcionales para el desarrollo del módulo, así como el modelado de la solución a partir de dichos requisitos. Luego de la implementación del módulo propuesto se realizan las pruebas a las funcionalidades más importantes, permitiendo validar la tecnología y el modelado propuesto mediante los resultados satisfactorios de las pruebas.

Palabras clave: GESPRO, gestión de activos, módulo, reutilización .

Índice general

Índice de figuras

Índice de tablas

Lista de códigos fuentes

Introducción

La gestión de proyectos es una disciplina que ha evolucionado desde sus orígenes en la ingeniería y construcción de proyectos. Las organizaciones, con el uso de esta disciplina, han sido capaces de identificar la necesidad de cambio, de diseñar las modificaciones necesarias y de implementarlas con eficacia y eficiencia.

En la actualidad existen varias herramientas para la gestión de proyectos tales como Redmine, phpCollab, DotProject, Trac, Microsoft Project, entre otras, que a pesar de tener ciertas funcionalidades implementadas, estas no suplen aún todas las áreas del conocimiento de la gestión de proyectos. Basado en lo anteriormente planteado, surge como iniciativa, el desarrollo de la herramienta GESPRO para la Universidad de las Ciencias Informáticas (UCI), que permite suplir las necesidades de los diferentes centros de producción de la institución, en el marco de la disciplina de Gestión de Proyectos.

GESPRO es una suite que se presenta en un modelo de negocios basado en servicios que combinan el uso de una solución informática para la Dirección Integrada por Proyectos (DIP) y un sistema de formación especializada en gestión de proyectos (QUIÑONES, 2014). Esta combinación posibilita no sólo la informatización de las organizaciones sino también la mejora integral de los procesos de planificación, control y seguimiento de proyectos. Es por ello que la UCI ha asumido la suite GESPRO como sistema base para la gestión de proyectos en su red de centros de desarrollo de soluciones y servicios informáticos, ya que dicha suite cubre todas las áreas del conocimiento de la gestión de proyectos.

Una de las áreas del conocimiento de la gestión de proyectos es la Gestión Logística y de Adquisiciones la cual incluye los procesos necesarios para comprar o adquirir productos, servicios o resultados que es preciso obtener fuera del equipo del proyecto (PMI, 2013). Dentro de esta área, GESPRO permite la gestión de adquisiciones hasta la asignación a proyectos, la gestión logística, contrato con proveedores así como la gestión del Fondo de Recursos Compartidos (FCR, por sus siglas en inglés).

El FCR es la base de la logística y el control de la ejecución de los proyectos (LÓPEZ, 2012). Si tomamos en cuenta que los recursos humanos, equipos, productos de software, materiales y de financiamiento constituyen la base para la formación del FCR, entonces podemos asegurar que muchos de estos recursos, a excepción de los recursos humanos, constituyen activos de la organización, sobre los cuales actúan el comercio, la planificación, las compras, la contabilidad, la economía, los talleres y los departamentos especializados de la estructura funcional, por lo que se hace viable la gestión de estos activos de la organización para obtener un valor de los mismos.

La gestión de activos es un conjunto de procesos capaz de garantizar el control eficaz de los activos de una organización. La misma, unida a un sistema de gestión, puede aportar beneficios y aprovechamiento

de nuevas oportunidades a las organizaciones para obtener valor de sus activos en el logro de los objetivos previstos, cuyo valor dependerá de dichos objetivos, del propósito de la organización, de las necesidades y expectativas de sus partes interesadas. La correcta gestión de activos, puede proveer de beneficios notables a las organizaciones, tales como la mejora del desempeño financiero, la reputación, la eficiencia, la eficacia, y la reutilización de los activos, entre otros (ISO, 2014).
La reutilización es la acción de reciclar los bienes o productos de una organización y constituye una forma más económica de producir con mayor rapidez y calidad. Es por ello, que si se catalogan los activos como útiles de una organización, entonces se puede afirmar que los mismos pueden ser reutilizables.

El control de los activos de la UCI es de vital importancia para el desarrollo de proyectos y soluciones más eficientes y económicas. Al tener un registro de los activos de la Universidad se puede realizar un registro de los costos de producción de los mismos, así como del valor asociado de cada activo en los diferentes proyectos donde se han utilizado. La UCI a pesar de ser uno de los mayores centros productores de software del país, no siempre compra nuevas tecnologías sino que utiliza el fondo de activos de la Universidad. Actualmente en el fondo de activos de la institución, solo se contemplan los activos tangibles o físicos, y no se tienen en cuenta los activos intangibles, específicamente los de software, que constituyen los componentes de software que se producen en la entidad. Estos componentes de software, como no están registrados como activos fijos de la Universidad, no se pueden emplear como recursos para la elaboración de nuevos proyectos. Debido a esto, la UCI, a la hora de estimar su valor, solo toma en cuenta su infraestructura, es decir los activos tangibles, cuando realmente en las organizaciones orientadas al desarrollo de software su valor real está dado, principalmente, por los activos de software que se desarrollan; esto conlleva al poco aprovechamiento de los recursos intangibles de la Universidad.

La UCI presenta una red de producción a través de centros de desarrollo orientados a proyectos, unidos según el campo de aplicación del software que desarrollan, siendo muy similares las características de los software y entornos de aplicación empleados en estos centros (PUPO, 2011). Sin embargo, no siempre se crean condiciones favorables para un entorno de reutilización, debido fundamentalmente a que no se dispone de un sistema para la gestión de activos tangibles e intangibles que potencie la reutilización, una adecuada especificación y control de los activos con calidad, lo cual puede generar pérdidas a la institución.

La suite GESPRO contiene diferentes proyectos asociados a los centros de producción de la UCI, estos proyectos tienen tareas definidas, estas a su vez necesitan recursos para su respectivo cumplimiento, los cuales se encuentran en el FCR o se compran. Para ello, la suite, presenta un módulo de gestión de recursos, que registra las compras de los recursos asociados a un proyecto o centro de desarrollo, además permite estimar la cantidad de recursos que se necesitan para la elaboración de un proyecto así como la estimación de sus precios. No obstante, no permite asignar a un proyecto, un recurso a utilizar que se encuentre en un área específica de la institución. Por tal motivo, GESPRO no resuelve el problema de asignación de recursos a los proyectos.

Actualmente, en la suite GESPRO 14.05 existe un módulo que permite registrar, hacer búsquedas y análisis de los activos intangibles; sin embargo, no gestiona activos tangibles y presenta insuficiencias respecto

a otras funcionalidades que también se pueden ofrecer, tales como, el cálculo de la amortización acumulada y el valor actual de un activo registrado en el sistema.

Luego de analizar la problemática anteriormente descrita se identifica como **problema de la investigación**: ¿Cómo gestionar los activos de una organización orientada a proyectos?.

La investigación en curso enmarca como **objeto de estudio**: la gestión de recursos reutilizables, delimitando como **campo de acción**: la gestión de activos en GESPRO.

Para dar solución al problema de la investigación se define como **objetivo general**: diseñar e implementar un módulo para la gestión de activos para la versión 16.05 de GESPRO.

Se desglosan del objetivo general los siguientes **objetivos específicos:**

- Construir un marco teórico asociado a las soluciones de GESPRO y a los procesos de gestión de activos.
- Diseñar un módulo para la gestión de activos para la versión 16.05 de GESPRO.
- Implementar y validar el módulo a desarrollar para la gestión de activos para GESPRO 16.05.

Posibles resultados:

- Especificación de requisitos de software.
- Historias de usuarios.
- Código fuente.
- Diseños de casos de pruebas.
- Prototipo funcional del módulo.

Tareas de investigación:

- Caracterización de los procesos relacionados con la gestión de los activos.
- Descripción del estado del arte de la herramienta existente en GESPRO para la gestión de activos.
- Selección y argumentación de las tendencias y tecnologías actuales a utilizar en el proceso.
- Identificación de los requisitos funcionales y no funcionales.
- Elaboración del modelo de diseño e implementación.
- Implementación de las historias de usuarios definidas a partir de los requisitos funcionales.
- Aplicación de las pruebas al módulo.

Idea a defender: Si se desarrolla un módulo para la gestión de activos para la Suite GESPRO 16.05, entonces se facilita la reutilización y gestión de los activos de los proyectos.

Para darle cumplimiento a las tareas se guía la investigación en los marcos de los **métodos científicos teóricos y empíricos** .

Métodos teóricos:

Analítico-sintético: para analizar la documentación relacionada con los contenidos referentes a la gestión de activos existentes, lograr una mayor comprensión y sintetizar información consultada para el desarrollo de la investigación.

Histórico-lógico: para analizar la evolución y desarrollo de los métodos y herramientas empleadas en el proceso de instalación y configuración de la herramienta GESPRO.

Modelación: para la elaboración de los artefactos generados durante el análisis y el diseño de la solución propuesta.

En la investigación también se utiliza el **método empírico**:

Observación: para constatar las debilidades y fortalezas existentes en la gestión de activos en la herramienta GESPRO 14.05.

El documento de tesis consta de tres capítulos que a su vez se dividen en epígrafes y estos en subepígrafes de acuerdo al nivel de detalle que requiere el contenido abordado en cada uno de ellos. A continuación se explica brevemente el contenido de cada capítulo:

Capítulo I. Fundamentación teórica. Recoge el análisis de la información existente acerca del tema a tratar. También incluye una descripción que da un sustento teórico a la selección de las herramientas y la metodología a utilizar para resolver el problema planteado.

Capítulo II. Propuesta de solución. Descripción de los requisitos y análisis de la solución propuesta. Se describen las historias de usuarios que modelan la solución y sus principales funcionalidades.

Capítulo III. Implementación y pruebas. Se ofrecen los detalles relacionados con la implementación de las funcionalidades definidas en el capítulo anterior y la validación de la solución propuesta, refiriéndose al diseño de las pruebas.

CAPÍTULO 1

Fundamentación teórica

1.1. Introducción

En el presente capítulo se realiza una investigación para el análisis y comprensión de los principales conceptos relacionados con la investigación. Además se describen las tendencias presentes en los sistemas involucrados en el proceso investigativo. Se justifican las herramientas, tecnologías y metodologías, así como el frameworks, lenguaje de programación, la plataforma y el servidor de aplicación que se utilizan en el desarrollo de éste trabajo para dar solución al problema de la investigación.

Conceptos asociados a la investigación

Seguidamente se exponen los principales conceptos relacionados con el objeto de estudio para facilitar el desarrollo del módulo propuesto. Los mismos guían la investigación que da solución al problema a resolver.

1.2. Organización orientada a proyectos

Según el PMBOK existen diferentes tipos de estructuras organizacionales entre las que se destacan: las estructuras funcionales, las estructuras matriciales (débiles, equilibradas o fuertes) y las estructuras orientadas a proyectos. Por otro lado, aquellas entidades cuyo objeto social está orientado al desarrollo de proyectos generalmente adoptan una estructura orientada a proyectos y se les llama comúnmente organizaciones orientadas a proyectos (PMI, 2013).

En las organizaciones orientadas a proyectos, los miembros del equipo suelen estar trabajando en el mismo lugar físico con directores de proyecto con gran independencia y autoridad. Este tipo de estructuras se observa en organizaciones que obtienen sus ingresos principalmente de proyectos (LLEDÓ, 2015). Estas organizaciones tienden a utilizar la gestión de proyectos mediante un sistema de gestión para facilitar la dirección de proyectos.

1.3. Gestión de proyectos

La gestión de proyectos implica la planificación, supervisión y control del personal, del proceso y de los eventos que ocurren mientras evoluciona el software desde la fase preliminar hasta la implementación operacional (R. S. PRESSMAN, 2002).

Según Sommerville: la gestión de proyectos de software es una parte esencial de la ingeniería del software. La buena gestión no puede garantizar el éxito del proyecto; sin embargo la mala gestión usualmente te lleva al fracaso del proyecto (SOMMERVILLE y GALIPIENSO, 2005).

La gestión de proyectos tiene como finalidad principal la planificación, el seguimiento, el control de las actividades, de los recursos humanos y materiales que intervienen en el desarrollo de un Sistema de Información. Como consecuencia de este control es posible conocer en todo momento qué problemas se producen y resolverlos o paliarlos de manera inmediata (AJENJO, 2005).

1.4. Gestión logística y de adquisiciones

Dentro de las áreas del conocimiento de la Gestión de Proyectos se encuentra la Gestión Logística y de Adquisiciones, la cual, además de incluir los procesos necesarios para comprar o adquirir productos, servicios o resultados se encarga de *los procesos de gestión del contrato y de control de cambios requeridos para desarrollar y administrar contratos u órdenes de compra emitidos por miembros autorizados del equipo del proyecto* (PMI, 2013).

En el área de la Gestión de las Adquisiciones del proyecto se incluyen varios procesos (ver anexo A.1) que involucran acuerdos, incluidos los contratos, que son documentos legales que se establecen entre un comprador y un vendedor.

1.4.1. Fondo de recursos compartidos

El área del conocimiento de la Gestión Logística y de Adquisiciones contiene un FCR, cuya base la conforman los recursos humanos, equipos, productos de software, materiales y de financiamiento, sobre los cuales actúa el ámbito comercial, de planificación, compras, contabilidad, economía, etc; siendo una de las partes fundamentales de esta área ya que puede garantizar la integración de una organización en función de su mejora.

Por otro lado, los objetivos de los proyectos responden a planes estratégicos de la organización y aunque estos sean diferentes, los recursos son comunes a los mismos, por lo que si se comparten, se facilita la integración mediante un FCR que optimiza su uso a partir de tener definidas las prioridades de los proyectos (LÓPEZ, 2012).

Una de las mejores formas de trabajar cuando se tienen varios proyectos en una organización es mediante la creación de un fondo de recursos compartidos, sobre el cual van a incidir todos los proyectos; de esta forma se pueden realizar correctamente los balances, las curvas de suministro y los pedidos para garantizar

la distribución de los recursos en función de la programación de los proyectos.

Muchos de estos recursos compartidos pueden ser catalogados como activos de la organización (a excepción de los Recursos Humanos), ya que algunos de ellos, pueden aportar un valor a la misma; por lo que se hace necesario una correcta gestión de activos para tener un control de los mismos.

1.5. Gestión de activos

La Norma ISO 55000 para la gestión de activos (ISO, 2014) dentro de su contenido, define como gestión de activos: *actividad coordinada de una organización para obtener valor a partir de los activos*. Dependiendo del valor obtenido de cada activo, generalmente implica balance de costos, riegos, oportunidades y beneficios de desempeño.

La Norma ISO 55000 define varios de los factores que influencian el tipo de activos que requiere una organización para alcanzar sus objetivos y cómo se gestionan los activos, tales como:

- la naturaleza y propósito de la organización;
- su contexto operacional;
- sus restricciones financieras y los requisitos reglamentarios;
- las necesidades y expectativas de la organización y sus partes interesadas.

Es fundamental considerar estos factores influyentes al establecer, implementar, mantener y mejorar consecutivamente la gestión de los activos. La organización debe garantizar un eficaz control y gobernanza de activos para alcanzar un valor de los mismos con el fin de lograr un balance deseado entre el costo, riesgo y desempeño. La organización debe de proporcionar un repositorio para la consulta y el almacenamiento de los activos en el transcurso de su vida útil.

1.6. Repositorio de activos

Según la Real Academia de la Lengua Española (RAE), el término repositorio refiere a : *lugar donde se almacena algo*. En la informática, repositorio, responde a un sistema de almacenamiento, que contiene grandes cantidades de información. Repositorio de activos refiere al sistema de información para guardar y gestionar tanto los activos de reutilización como la información que se almacenen de ellos, tal como: la descripción general de cada activo. Su objetivo fundamental es asegurar la disponibilidad de componentes para apoyar la ingeniería de aplicaciones y facilitar la visualización y reutilización por los miembros de la red. Al ser una concentración de activos debe ser asegurado y protegido (JORRÍN, 2012).

1.7. Reutilización

El término reutilizar: *es volver a utilizar algo, generalmente con una función distinta a la que tenía originariamente* (RAE, 2016b).

Según Sommerville: la ventaja obvia de la reutilización de software es que los costes totales de desarrollo deben reducirse, sin embargo la reducción de costes es sólo una ventaja de la reutilización (SOMMERVILLE y GALIPIENSO, 2005).

Peter Freeman plantea que el propósito de la reutilización *es mejorar la eficiencia, la productividad y la calidad del desarrollo de software. Así la reutilización puede definirse como cualquier procedimiento que produce o ayuda a producir un sistema mediante el nuevo uso de algún elemento procedente de un esfuerzo de desarrollo anterior* (FREEMAN, 1987).

La reutilización constituye uno de los principales temas de investigación en el campo de la Ingeniería del Software, citándose frecuentemente como una de las principales técnicas para incrementar la productividad de los desarrolladores de software (ALLEN, 2014).

Desde el inicio del presente siglo ha emergido como elemento central de la reutilización, el concepto de activos reutilizables, que no es más que un artefacto, componente e incluso un bien tangible que puede ser utilizado múltiples veces en el transcurso de su vida útil (HENRIK PESTANO PINO, 2011).

1.8. Activo

Un activo es algo que posee valor potencial o real para una organización. El valor puede variar entre diferentes organizaciones y sus partes interesadas y puede ser tangible o intangible (ISO, 2014).

Se define como período de un activo, desde la generación de un activo, a la creación o generación del mismo hasta el final de su vida como activo. La vida del activo no necesariamente debe de coincidir con el período durante el cual una organización tiene responsabilidad sobre este; sin embargo, un activo puede suministrar un valor potencial o real a una o más organizaciones en el transcurso de su vida, no obstante, el valor del activo a lo largo de su vida, puede cambiar para una organización en cualquier momento dependiendo de la frecuencia con que deprecie y de su tasa de depreciación, aún cuando deje de ser útil para la organización tiene siempre un valor residual asociado.

1.8.1. Activo físico e intangible

Activos físicos o tangibles generalmente se refieren a equipamiento, inventario y los inmuebles de la organización. Los activos intangibles son lo opuesto a los activos físicos, pueden ser alquileres, marcas, activos digitales, derechos de uso, licencias, derechos de propiedad intelectual, reputación o acuerdos, etc.

Dentro de los activos intangibles se encuentran los activos de software, este término abarca todos los artefactos que se generan de un software y que pueden ser reutilizados,es decir, constituye un producto de software que puede ser utilizado en diversas ocasiones en el desarrollo de diferentes aplicaciones o sistemas. Según (PUPO, 2011), un activo de software puede ser:

- Un componente de software
- Una especificación de requisitos
- Un modelo de negocios

- Una especificación de diseño
- Un algoritmo
- Un patrón de diseño
- Una arquitectura de dominio
- Un esquema de base de datos
- Una especificación de prueba
- La documentación de un sistema
- Un plan

1.8.2. Activo de software reutilizables

Se puede definir como activo de software reutilizable al producto de software diseñado expresamente para ser utilizado múltiples veces en el desarrollo de diferentes sistemas, aplicaciones o productos.

1.9. Depreciación y amortización

La RAE define al término **depreciación**: *disminución del valor o precio de algo, ya con relación al que antes tenía, ya comparándolo con otras cosas de su clase* (RAE, 2016a).

La depreciación es el mecanismo mediante el cual se reconoce el desgaste que sufre un bien por el uso que se haga del mismo. Cuando un activo es utilizado para generar ingresos, este sufre un desgaste normal durante su vida útil que al final lo lleva a ser inutilizable. Al ingreso generado por el activo usado, se le debe incorporar el gasto correspondiente de desgaste que ese activo sufre para poder generar el ingreso, según señala un elemental principio económico, no puede haber ingreso sin haber incurrido en un gasto, y el desgaste de un activo por su uso, es uno de los gastos que al final permiten generar un determinado ingreso (GERENCIE, 2016).

En la actualidad, el concepto de depreciación solo es asociado a los activos físicos ya que estos tienden a perder valor para una organización con el paso del tiempo, no obstante para el caso de los activos intangibles se utiliza el término **amortización**.

La amortización es la reducción del valor o "desgaste" de los activos intangibles con el paso del tiempo y esa pérdida se amortiza teniendo en cuenta el tiempo de vida útil del activo.

Por ejemplo, si se desarrolla un activo de software con cierta tecnología, luego de pasados unos años, dicha tecnología se vuelve obsoleta y por ende el activo desarrollado por la misma es obsoleto, por lo que su amortización puede ser calculada tomando en cuenta la tecnología con la que fue creado. Por tanto, se puede concluir que un activo de software se le puede asociar el término **amortización** ya que con el paso del tiempo también amortiza según el desarrollo de las tecnologías.

Lingüísticamente hablando de la amortización de activos, el término "depreciación" probablemente es más apropiado para reflejar el ajuste del valor. No obstante, en el ambiente de contabilidad internacional (Normas Internacionales de Contabilidad (NIC) , Normas Internacionales de Información Financiera (NIIF))

las dos palabras significan cosas diferentes. Por tal motivo, el uso de "amortización" se limita a activos intangibles y el de "depreciación" se refiere a activos físicos (DEBITOOR, 2016).

Para el desarrollo de la investigación se toman los conceptos **depreciación** y **amortización** teniendo en cuenta lo anteriormente planteado.

1.9.1. Métodos de depreciación/amortización

Para poder estimar o calcular un valor de los activos registrados en un sistema, se emplean varios métodos de depreciación/amortización; la selección de los métodos se realiza teniendo en cuenta el escenario en que se encuentre el activo. Estos métodos son aplicables tanto para los activos físicos como los intangibles. A continuación se mencionan algunos de los métodos de depreciación/amortización más utilizados:

1. Método de la línea recta.
2. Método de actividad o unidades producidas.
3. Método de la suma de dígitos anuales.
4. Método de doble cuota sobre el valor que decrece.

Para la investigación propuesta se utiliza el **método de doble cuota sobre el valor que decrece**, respetando los requisitos del cliente para el desarrollo de la misma. En dicho método básicamente se definen las siguientes fórmulas:

$$VA^1 = Costo - DA^2$$

$$DA = DAA^3 + (Costo * TD^4)$$

1.9.2. Depreciación acumulada

La depreciación/amortización acumulada es un concepto que se maneja dentro del mundo de la contabilidad financiera, en el que la parte que corresponde al término **acumulada** se refiere al período de tiempo que tarda un activo en depreciarse. La cuenta depreciación/amortización acumulada es una cuenta compensatoria que reduce o disminuye la cuenta de activos fijos. Esta cuenta no se cierra al terminar el período contable, por el contrario, continúa aumentando hasta que el activo se haya depreciado por completo, vendido o dado de baja (DEPRECIACIÓN, 2016).

1.9.3. Tasa de depreciación/amortización

La Tasa de Depreciación corresponde a los porcentajes de valor que sufre cada activo de manera mensual o anual; por tal motivo el costo de los activos decrece en función de estas tasas de depreciación/amortización, y los activos se van deteriorando o desgastando, por lo que durante su vida útil pierden parte de su valor.

[1] Valor actual
[2] Depreciación/amortización acumulada
[3] Depreciación/amortización acumulada actual
[4] Tasa de depreciación/amortización

Ejemplos de tasas de depreciación:

- Silla: la tasa de depreciación/amortización de una silla, asciende al 2 % mensual.
- Maquinaria y mobiliario: la tasa de depreciación de este tipo de activos corresponde a un 10 % anual.

1.10. Valor residual

El valor residual de un activo es el importe que, en el momento actual, se estima que la organización podría obtener por su venta u otra forma de disposición, una vez deducidos los costes estimados para realizar la venta, al final de su vida útil (CONTABLE, 2016).

El valor residual de un activo no es más que el valor final del mismo, una vez que haya perdido su valor, tras haber sido utilizado durante unos años de vida determinados. Consiste en un cálculo de estimación de cuál es su valor en el momento en que ya no se utilice (DEBITOOR, 2016).

1.10.1. Vida útil

Es el período de tiempo durante el cual la organización espera utilizar el activo amortizable, o el número de unidades de producción que se espera obtener del mismo (CONTABLE, 2016).

1.11. Módulo

A la hora de estructurar el código de cualquier tipo de programa es esencial reconocer aquellas secuencias que se utilizan más de una vez, para evitar la repetición innecesaria de líneas. Agrupar aquellas tareas que se realizan con frecuencia en funciones no sólo ofrece el beneficio inmediato de volver el código más prolijo y legible, sino que reduce considerablemente el tamaño de una aplicación.

En términos informáticos, diríamos que se divide un programa en una serie de subprogramas, que en este caso son los módulos. Como en otros campos, si uno de los módulos presenta algún comportamiento inesperado, resulta fácil detectarlo y trabajar en él sin afectar al resto. Además, las grandes organizaciones suelen asignar un mismo proyecto a decenas de programadores, que en muchos casos superan los 100, y la repartición del trabajo, además del diseño modular, resulta la forma más inteligente de encarar el desarrollo.

Ian Sommerville afirma que *un módulo es normalmente un componente de un subsistema que proporciona uno o más servicios a otros módulos. A su vez éste usa los servicios proporcionados por otros módulos* (SOMMERVILLE y GALIPIENSO, 2005). Es decir, un módulo no se considera un sistema independiente, sino que normalmente se componen de varios componentes del sistema mucho más simples.

1.12. Módulo para la gestión de activos de GESPRO 14.05

GESPRO 14.05, cuenta con un módulo para la gestión de Alcance Trazabilidad, dentro del cual, existen funcionalidades asociadas a los activos de software y los requisitos de la organización, permitiendo la gestión

de los mismos, así como la asociación entre ellos. El módulo, actualmente es el encargado de administrar y gestionar los activos y requisitos de los proyectos de desarrollo vinculados a la herramienta GESPRO 14.05, dicho módulo, fue diseñado e implementado bajo las especificaciones técnicas de la herramienta antes mencionada, el mismo está dividido en dos niveles, a nivel de centro y de proyecto, presentando funcionalidades que facilitan la gestión de los activos.

En la presente investigación solo se tienen en cuenta las funcionalidades asociadas a la gestión de activos que ofrece el módulo de Alcance Trazabilidad, ya que los requisitos de la organización no presentan relación con los activos físicos por lo que puede generar incongruencias a lo largo de la investigación.

A continuación se realiza una descripción de las funcionalidades basadas en la gestión de activos que presenta el módulo antes mencionado:

- **Nivel de centro**
  - **Gestión de activos de software**: permite realizar las operaciones básicas mediante un crear, obtener, actualizar, eliminar (CRUD, por sus siglas en inglés).
  - **Asociar activos**: permite asociar los activos registrados en el sistema, así como eliminar dicha asociación si fuera necesario.
  - **Asociar a requisitos**: permite asociar un activo con varios requisitos registrados en el sistema.
  - **Filtrar por características**: permite el filtrado de activos por procesos, categorías y estados.
  - **Buscar activo de software**: permite la búsqueda de activos en el sistema según el nombre del activo, descripción y palabras claves que se le definen en su entrada al sistema.

- **Nivel de proyecto**
  - **Gestión de activos de software**: permite realizar las operaciones básicas CRUD.
  - **Asociar activos**: permite asociar los activos registrados en el proyecto, así como eliminar dicha asociación si fuera necesario.
  - **Asociar a requisitos**: permite asociar un activo con varios requisitos registrados en el proyecto.
  - **Filtrar por características**: permite el filtrado de activos por procesos, categorías y estados.
  - **Buscar activo de software**: permite la búsqueda de activos en el sistema según el nombre del activo, descripción y palabras claves que se le definieron en su entrada al sistema.

1.12.1. Ventajas del uso del módulo Alcance Trazabilidad respecto a la gestión de activos

Dadas las funcionalidades definidas anteriormente se puede concluir que el módulo Alcance Trazabilidad definido para la suite GESPRO 14.05 provee de beneficios notables a la organización tales como:

- Gestión y control de los activos de software registrados en el sistema.
- Garantiza la reutilización de los activos de software registrados en los proyectos vinculados a la suite.
- Vincula los requisitos definidos en el centro o de un proyecto en específico con los activos de software registrados respectivamente.

1.12.2. Deficiencias del módulo Alcance Trazabilidad respecto a la gestión de activos

El módulo de Alcance Trazabilidad a pesar de proporcionar ventajas a la herramienta GESPRO en su versión 14.05, no es ajeno a deficiencias respecto a la gestión de activos, tales como:

- No permite la gestión de los activos físicos de la organización
- No incluye características asociadas a los activos físicos de la organización.
- No utiliza conceptos relacionados con los activos tangibles e intangibles, tales como: depreciación, tasa de depreciación y valor actual del activo.
- No permite estimar un valor real de un activo con el paso del tiempo.

1.13. Selección de los lenguajes, tecnologías y herramientas a utilizar

Para el desarrollo de la investigación se utilizan las herramientas, tecnologías y lenguajes de programación definidos por el cliente, buscando respetar la uniformidad de la suite GESPRO. A continuación se describen las seleccionadas.

1.13.1. Metodologías ágiles

Las metodologías ágiles combinan una filosofía y conjunto de directrices de desarrollo. La filosofía busca la satisfacción del cliente y la entrega temprana de software incremental, equipos de proyecto pequeños y con alta motivación, métodos informales, un mínimo de productos de trabajo de la ingeniería de software, y una simplicidad general de desarrollo. Las directrices de desarrollo resaltan la entrega sobre análisis y diseño (aunque estas actividades no se descartan) y la comunicación activa entre los desarrolladores y los clientes (R. PRESSMAN, 2005).

Entre las metodologías ágiles más destacadas hasta el momento se pueden nombrar:

- Extreme Programming (XP).
- Scrum.
- Crystal Clear.
- Dynamic Systems Developmemt Method (DSDM).
- Feature Driven Development (FDD).
- Adaptive Software Development (ASD).
- XBreed.
- Extreme Modeling.

Las metodologías ágiles están especialmente indicadas para proyectos con requisitos poco definidos o cambiantes. Estas se aplican bien en equipos pequeños que resuelven problemas concretos, lo que no está reñido con su aplicación en el desarrollo de grandes sistemas, ya que una correcta modularización de los mismos es fundamental para su exitosa implantación. Dividir el trabajo en módulos abordables minimiza los fallos y el coste (ALLEN, 2014).

1.13.2. Metodología de desarrollo de software: SCRUM

SCRUM es una metodología ágil de desarrollo de proyectos que toma su nombre y principios de los estudios realizados sobre nuevas prácticas de producción por Hirotaka Takeuchi e Ikujijo Nonaka a mediados de los 80 (PALACIO, 2006). Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años.

Según (PALACIO, 2014): SCRUM es un modelo de desarrollo ágil caracterizado por:

- Adoptar una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto.
- Basar la calidad del resultado más en el conocimiento tácito de las personas en equipos auto-organizados, que en la calidad de los procesos empleados.
- Solapamiento de las diferentes fases del desarrollo, en lugar de realizar una tras otra en un ciclo secuencial o de cascada.

Según (GALLEGO, 2015), SCRUM es adecuado para aquellas organizaciones en las que el desarrollo de los productos se realiza en entornos que se caracterizan por tener:

1. Incertidumbre: sobre esta variable se plantea el objetivo que se quiere alcanzar sin proporcionar un plan detallado del producto. Esto genera un reto y da una autonomía que sirve para generar una "tensión" adecuada para la motivación de los equipos.
2. Auto-organización: los equipos son capaces de organizarse por sí solos, no necesitan roles para la gestión pero tienen que reunir las siguientes características:
    - Autonomía: son los encargados de encontrar la solución usando la estrategia que encuentren adecuada.
    - Auto-superación: las soluciones iniciales sufrirán mejoras.
    - Auto-enriquecimiento: al ser equipos multidisciplinares se ven enriquecidos de forma mutua, aportando soluciones que puedan complementarse.
3. Control moderado: se establece un control suficiente para evitar descontroles. Se basa en crear un escenario de "autocontrol entre iguales" para no impedir la creatividad y espontaneidad de los miembros del equipo.
4. Transmisión del conocimiento: todo el mundo aprende de todo el mundo. Las personas pasan de unos proyectos a otros y así comparten sus conocimientos a lo largo de la organización.

Según (PALACIO, 2014), para un entendimiento de las fases (ver anexo A.2) se debe de tener un conocimiento previo de los siguientes **conceptos:**

- Pila del producto: (product backlog) lista de requisitos de usuario, que a partir de la visión inicial del producto crece y evoluciona durante el desarrollo.
- Pila del sprint: (sprint backlog) lista de los trabajos que debe realizar el equipo durante el sprint para generar el incremento previsto.

- Sprint: nombre que recibe cada iteración de desarrollo. Es el núcleo central que genera el pulso de avance por tiempos prefijados (time boxing).
- Incremento: resultado de cada sprint.

Luego del análisis de las características anteriores se decide que SCRUM es la metodología idónea para dirigir el proceso de desarrollo del módulo asociado a la investigación, puesto que es una metodología de desarrollo muy simple y por tal motivo está dirigida a proyectos con requisitos inestables y que requieren rapidez y flexibilidad, además de ser una metodología adecuada cuando se tienen todas las especificaciones de forma inicial, el tipo de producto ya es conocido o es un proyecto que se entiende su naturaleza perfectamente, lo cual se asemeja a la forma de trabajo del equipo de proyecto de GESPRO. Es importante destacar, que todos lo productos pertenecientes a la suite GESPRO, están basados en dicha metodología y por ende es un requisito del cliente el empleo de esta metodología para el desarrollo de la solución propuesta.

1.13.3. Lenguaje para modelado: UML

El Lenguaje de Modelado Unificado (UML, por sus siglas en inglés) es un lenguaje basado en diagramas para la especificación, visualización, construcción y documentación de cualquier sistema complejo. (FUOC, 2014)

UML se ha convertido en ese estándar tan ansiado para representar y modelar la información con la que se trabaja en las fases de análisis y, especialmente, de diseño. El lenguaje UML tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático: desde el análisis con los casos de uso, el diseño con los diagramas de clases, objetos, etc., hasta la implementación y configuración con los diagramas de despliegue. (ORALLO, 2014)

Según la definición de los creadores del modelo es una “simplificación de la realidad” que proporciona los planos de un sistema. Así pues, los diagramas UML además de una forma de documentar un proyecto, son sobre todo, una herramienta que nos ayuda en el proceso de abstracción y definición de las funcionalidades del sistema. (CERDA, 2003)

UML es independiente de las metodologías de análisis y diseño y de los lenguajes de programación que se utilicen en la construcción de los sistemas software, es importante destacar que se basa en el paradigma de la orientación a objetos. Por tanto, es especialmente adecuado para la construcción de sistemas software desde la perspectiva de la orientación a objetos. (FUOC, 2014)

1.13.4. Lenguaje de programación: Ruby 2.2.2

El lenguaje Ruby fue inventado por Yukihiro “Matz” Matsumoto, en Japón, en 1995. Siguiendo la tradición de los lenguajes de programación que han sido desarrollados recientemente, y que gozan de popularidad, Ruby es un lenguaje interpretado, gratuito (Open Source), y orientado por objeto. “Matz” admite que se inspiró en los lenguajes Perl y Python, pero Ruby es mucho más orientado por objeto; de hecho, todo en Ruby son objetos. Es un lenguaje genérico que se puede utilizar en muchos campos: desde procesa-

miento de texto y programación web con CGI[5], hasta ingeniería, genética, y programación comercial a gran escala.(GUILLÉN, 2007)

Características generales del lenguaje:

- Orientado a objetos.
- Cuatro niveles de ámbito de variable: global, clase, instancia y local.
- Manejo de excepciones.
- Iteradores y clausuras o closures (pasando bloques de código).
- Expresiones regulares nativas similares a las de Perl a nivel del lenguaje.
- Posibilidad de redefinir los operadores (sobrecarga de operadores).
- Recolección de basura automática
- Altamente portable.
- Hilos de ejecución simultáneos en todas las plataformas usando greenthreads.
- Carga dinámica de DLL/bibliotecas compartidas en la mayoría de las plataformas.
- Introspección, reflexión y metaprogramación.
- Amplia librería estándar.
- Soporta inyección de dependencias.
- Soporta alteración de objetos en tiempo de ejecución.
- Continuaciones y generadores

(MATSUMOTO, 2001).

Para garantizar el proceso de implementación y diseño se elije este lenguaje en su versión 2.2.2 por las funcionalidades que ofrece, su fortaleza y dinamismo, además, de ser un lenguaje de código abierto enfocado en la productividad y simplicidad. Se toma también como un factor aún más decisivo que la plataforma GESPRO 14.04 fue implementada en dicho lenguaje, por tal motivo, todo componente que se desee agregar a dicha plataforma, debe de ser desarrollado en el mismo lenguaje de programación.

1.13.5. Framework Ruby on Rails 4.2.4 (RoR)

Se define como Framework a una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, se puede considerar como una aplicación genérica, incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta. Los objetivos principales que persigue un Framework son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones. Un Framework web, por tanto, se puede definir como un conjunto de componentes (por ejemplo clases en Java, descriptores y archivos de configuración en XML) que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas web.(GUTIÉRREZ, 2007)

[5]Interfaz de entrada común (en ingles Common Gateway Interface(CGI): importante tecnología de la World Wide Web (www) que permite a un cliente(navegador web) solicitar datos de un programa ejecutado en un servidor web

RoR es un Framework de desarrollo de aplicaciones web, de código abierto (open source, software libre) escrito en el lenguaje de programación Ruby, que permite construir aplicaciones web flexibles y robustas rápidamente.

1.13.6. Herramienta para el diseño: Visual Paradigm 8.0

Las herramientas Ingeniería de Software Asistida por Computadora (CASE, por sus siglas en inglés) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software, en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

Visual Paradigm es una herramienta CASE que permite la representación de los modelos en todas las dimensiones que UML abarca, y como todo programa computacional posee sus propios comandos de uso. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite construir todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación (PARADIGM, 2016).
Visual Paradigm también ofrece:

- Navegación intuitiva entre la escritura del código y su visualización.
- Potente generador de informes en formato PDF/Lenguaje de Marcas de Hipertexto (HTML, por sus siglas en inglés).
- Ambiente visualmente superior de modelado.
- Sofisticado diagramador automáticamente de layout.
- Sincronización de código fuente en tiempo real

(TARGETWARE, 2013).

1.13.7. Entorno de desarrollo IDE RubyMine 7.1.2

Se define como Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés), a la aplicación visual compuesto por un conjunto de herramientas para un programador, que sirve para la construcción de aplicaciones a partir de componentes. Consiste en un editor de código, un compilador, un depurador y en algunos casos un Constructor de Interfaz Gráfica (GUI, por sus siglas en inglés). Puede dedicarse o bien a un solo lenguaje de programación o poder utilizarse para varios de ellos. Un IDE debe tener las siguientes características:

- Multiplataforma.
- Soporte para diversos lenguajes de programación.
- Integración con sistemas de control de versiones.

- Reconocimiento de sintaxis.
- Extensiones y componentes para el IDE.
- Integración con frameworks populares.
- Depurador.
- Importar y exportar proyectos.
- Múltiples idiomas.
- Manual de usuarios y ayuda.

(ALLEN, 2014)

RubyMine es un IDE que permite producir un código de alta calidad de manera más eficiente, gracias al apoyo de primera clase para Ruby on Rails, Java Script y CoffeeScript, ERB y HAML, Hoja de estilo en cascada (CSS, por sus siglas en inglés), Sass entre otros.

1.13.8. PostgreSQL 9.4

PostgreSQL es un Sistema de Gestión de Bases de Datos Orientado a Objetos (SGBDOO) muy conocido y usado en entornos de software libre porque cumple los estándares SQL92 y SQL99, y también por el conjunto de funcionalidades avanzadas que soporta, lo que lo sitúa al mismo o a un mejor nivel que muchos Sistema de Gestión de Bases de Datos (SGBD) comerciales. (MARC GIBERT GINESTÁ, 2002)

Funciona en todos los principales sistemas operativos, incluyendo Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows. Dispone de interfaces de programación nativas para C / C++, Java, Net, Perl, Python, Ruby, Tcl, ODBC, entre otros, y una documentación excepcional. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multi-hilos para garantizar la estabilidad del sistema. (POSTGRESQL, 2016)

PgAdmin III es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia Open Source. Está escrita en C++ usando la librería gráfica multiplataforma wxWidgets, lo que permite que se pueda usar en Linux, FreeBSD, Solaris, Mac OS X y Windows. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3 ejecutándose en cualquier plataforma, así como versiones comerciales de PostgreSQL como Pervasive Postgres, EnterpriseDB, Mammoth Replicator y SRA PowerGres (UBUNTU, 2008). No se requieren controladores adicionales para comunicarse con la base de datos del servidor e incorpora funcionalidades para realizar consultas, examinar su ejecución y trabajar con los datos.

Conclusiones parciales

La definición de los conceptos de gestión de activos, activo, depreciación, amortización y reutilización, permite establecer el impacto sobre el cual recae la solución al problema de investigación planteada. El análisis de los principales conceptos relacionados a la investigación, brinda un mayor conocimiento sobre el campo de acción en el que se desarrolla la solución. El análisis del módulo Alcance Trazabilidad de la

versión 14.05 de GESPRO, posibilita determinar las principales fortalezas y debilidades del mismo en el marco de la gestión de activos, como ayuda a la investigación. La definición de la metodología SCRUM, para el desarrollo del módulo, permite ajustar las funcionalidades basándose en las necesidades del cliente. Las herramientas y tecnologías de desarrollo software, definidas por el cliente, permiten sentar las bases teóricas y tecnológicas para dar solución al problema planteado en la investigación.

CAPÍTULO 2

Propuesta de solución

2.1. Introducción

En el presente capítulo se realiza una descripción de la propuesta de solución relacionada a la investigación, conteniendo todo lo referente a la fase de análisis y modelado del diseño. Se capturan los requisitos funcionales del software que detallan las funcionalidades del módulo así como los requisitos no funcionales que son restricciones que debe cumplir el sistema. Además se describen los patrones arquitectónicos y de diseño a aplicar y se ilustran diagramas de clases del diseño, de componentes y de diseño de tablas de la base de datos.

2.2. Propuesta de solución

La presente investigación está orientada al desarrollo de un módulo que sea capaz de garantizar una correcta gestión y reutilización de los activos de los proyectos registrados en la Suite GESPRO 16.05, asegurando para cada proyecto poder obtener un valor de sus activos en el logro de sus objetivos. Supliendo también las carencias del vigente módulo de Alcance Trazabilidad de GESPRO 14.05, que permite la gestión de activos intangibles, específicamente de software, asociados a los requisitos de la organización. La solución propuesta además de permitir la gestión de todos los activos (físicos e intangibles) de la organización, elimina la asociación de los activos de los requisitos ya que estos últimos no presentan semejanzas con los activos físicos, por lo que puede generar incongruencias en el módulo. Luego del análisis de los parámetros antes mencionados, se define que la propuesta de solución aporta los siguientes beneficios:

1. **Mejora de la gestión de los activos intangibles:** consta con la inclusión de nuevas características y campos necesarios que describan mejor dichos activos.
2. **Gestión de activos físicos:** permite la gestión de todos los activos físicos de una organización.
3. **Cálculo de la depreciación/amortización acumulada:** a partir del costo y la tasa de depreciación/amortización asociados a un activo, se calcula la depreciación/amortización acumulada en tiempo

real.

4. **Cálculo del valor actual:** a partir del costo y de la depreciación/amortización acumulada asociados a un activo, se calcula su valor actual en tiempo real.
5. **Estimación del valor real de un activo:** permite realizar una estimación del valor real de un activo.
6. **Dependencia entre activos:** garantiza la asociación de los activos de forma más óptima.
7. **Empleo de la Norma ISO 55000 para la gestión de activos:** provee de aspectos generales de la gestión de activos, sus principios, terminologías y beneficios. Esta Norma Internacional puede aplicarse a todo tipo de activos y a cualquier tipo de organización sin importar el tamaño de esta.

En el módulo a desarrollar como propuesta de solución se definen los activos en dos grupos fundamentales: **físicos** o **intangibles**, para los cuales, dependiendo de la selección de uno de ellos, el sistema muestra sus respectivas características . Uno de los principales campos en común a llenar para ambos grupos es el **costo**, en el cual se inserta el costo del activo para la institución, de este dependen los campos **valor actual** y **depreciación/amortización acumulada**. El valor residual por su parte, debe ser tomado en cuenta al registrar el activo en la organización, pues este define el valor real del activo al terminar su vida útil en la organización. La solución propuesta es capaz de calcular en tiempo real, durante el primer día de cada mes, la depreciación/amortización acumulada de cada activo, y por ende su **valor actual** es modificado. Sin embargo, la depreciación/amortización acumulada es completamente diferente para cada activo dependiendo de sus valores de **costo** y **tasa de depreciación/amortización** introducidos. Para el cálculo de estas variables se utiliza el **método de la doble cuota sobre el valor decreciente**, ejecutando este último de manera automática, gracias al empleo de la gema[1] **whenever**.

2.2.1. Gema whenever

La gema whenever, es una gema de Ruby que permite crear tareas en Crontab de manera sencilla. Whenever está muy integrado al framework de desarrollo web Ruby on Rails, aunque se puede utilizar en un programa independiente que genere salidas de tareas para Crontab ya que la gema actualiza el archivo Crontab. Para el desarrollo del módulo propuesto, se emplea la gema whenever (ver anexo B.1), permitiendo al sistema realizar de manera automática el cálculo de la depreciación/amortización acumulada y el valor actual de los activos registrados de manera mensual, mediante la tarea (ver anexo B.1.1) definida en el módulo propuesto.

2.3. Modelo de Dominio

A la hora de obtener una mejor idea del análisis del problema de investigación se confecciona un modelo de dominio con el fin de mostrar a los usuarios de manera visual los principales conceptos que intervienen en el negocio y establece un lenguaje natural y común entre desarrolladores, clientes e interesados.

[1] Paquetes de librerías para Ruby que se instalan en el sistema y quedan listas para ser usadas

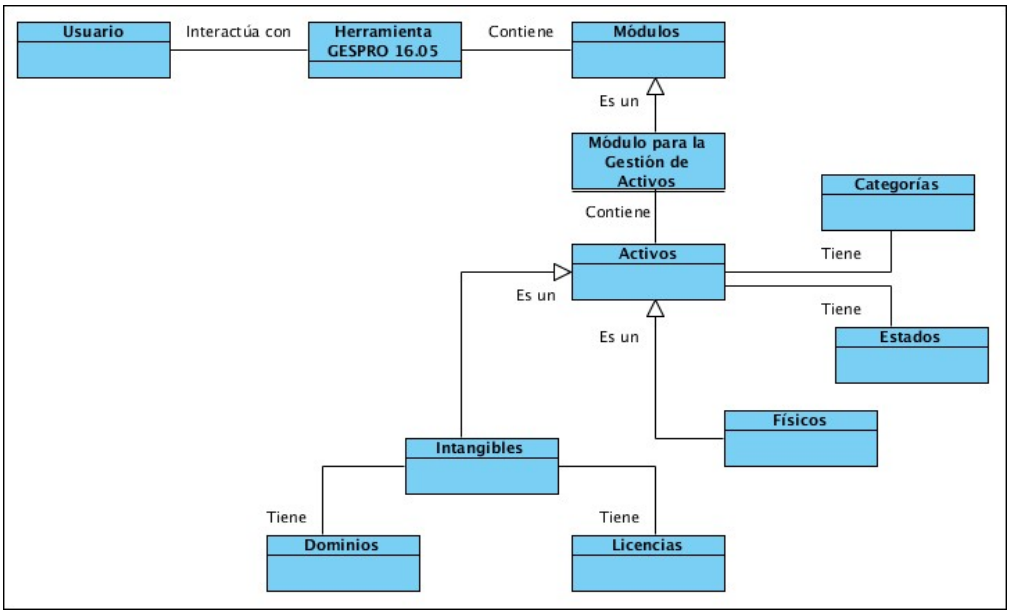


Figura 2.1. Modelo de dominio

Usuario: miembro de un proyecto determinado que en dependencia del rol que desempeñe y/o los permisos asociados al mismo en el módulo, puede utilizar las funcionalidades del mismo.

Herramienta GESPRO: herramienta para la gestión de proyecto extensible que incluye actualmente más de 100 reportes como apoyo al proceso de toma de decisiones en la gestión de proyectos y permite además la generación dinámica de nuevos reportes (GESPRO_UCI, 2013).

Módulos: un módulo es un software que agrupa un conjunto de subprogramas y estructuras de datos (ALEGSA, 2016).

Por otro lado, la Norma ISO 55000 para la gestión de activos (ISO, 2014) define los siguientes conceptos:

- **Módulo para la gestión de activos**: software para la gestión de activos cuya función es la de establecer la política de la gestión de activos y los objetivos de la misma.
- **Activos**: objeto o entidad que tiene valor real o potencial para una organización.
- **Activos físicos**: se refieren a equipamiento, inventario y los inmuebles de la organización.
- **Activos intangibles**: son activos no físicos como alquileres, marcas, activos digitales, derechos de uso, licencias, derechos de propiedad intelectual, reputación o acuerdos.

Categorías de activos: las categorías se definen según el tipo de activo, ejemplo:

- Activos físicos: equipamiento, inventarios e inmuebles.
- Activos intangibles: infraestructura, Tecnologías de la Información y la Comunicación (TIC), documentación técnica, documentación del cliente, etc.

Estados de Activos: los estados se definen según el tipo de activo, ejemplo:

- Activos físicos: En Explotación o En Uso.
- Activos intangibles: Definido, Elaborado, Certificado, Propuesto, Listo, Reutilizable, Publicado, Desactivado.

Dominios: los dominios solo se definen para el caso de los activos intangibles. Se pueden clasificar como: Gestión Empresarial, Sistemas de notificaciones, Seguridad, Marcos de trabajo, Herramientas y Telefonía.

Licencias: las licencias se definen solo para el caso de los activos intangibles. Se pueden clasificar como: Licencia Pública General (GPL, por sus siglas en inglés), Licencia Pública General Reducida (LGPL, por sus siglas en inglés) y Comercial.

2.4. Especificación de los requisitos del sistema.

La captura y escritura de los requisitos se refiere a la tarea de reunir una descripción del producto desde el punto de vista de negocio y es un paso importante para el proceso de desarrollo de un buen producto de software y debe tener en cuenta las exigencias por parte del cliente ya que estos suelen tener una noción de lo que quieren, una idea del resultado final; pero no de las funciones que deben llevarse a cabo para que su software sea como lo pensaron. Realizar con éxito este proceso de descripción de requerimientos ayuda a los ingenieros a comprender el problema y así desarrollar una solución que satisfaga las necesidades de los interesados. Los requisitos se pueden clasificar en dos grupos: Requisitos funcionales (RF) y Requisitos no funcionales (RNF) como se evidencia a continuación.

2.4.1. Requisitos funcionales.

Los RF de un sistema describen las capacidades o funciones que el sistema debe cumplir y los servicios que de él se esperan. Luego de un estudio de las necesidades que presenta el sistema en cuanto a la gestión de activos, se identificaron los siguientes RF:

RF1. Gestionar activos intangibles a nivel de centro
- **RF1.1.** Crear activo intangible a nivel de centro
- **RF1.2.** Modificar activo intangible a nivel de centro
- **RF1.3.** Eliminar activo intangible a nivel de centro
- **RF1.4.** Mostrar activo intangible a nivel de centro

RF2. Gestionar activos físicos a nivel de centro
- **RF2.1.** Crear activo físico a nivel de centro
- **RF2.2.** Modificar activo físico a nivel de centro
- **RF2.3.** Eliminar activo físico a nivel de centro
- **RF2.4.** Mostrar activo físico a nivel de centro

RF3. Filtrar activos a nivel de centro
- **RF3.1.** Filtrar por categorías de activos registrados a nivel de centro

RF3.2.. Filtrar activos por proyecto

RF3.3. Filtrar por estados de activos registrados a nivel de centro

RF4. Gestionar activos intangibles a nivel de proyecto

RF4.1. Crear activo intangible a nivel de proyecto

RF4.2. Modificar activo intangible a nivel de proyecto

RF4.3. Eliminar activo intangible a nivel de proyecto

RF4.4. Mostrar activo intangible a nivel de proyecto

RF5. Gestionar activos físicos a nivel de proyecto

RF5.1. Crear activo físico a nivel de proyecto

RF5.2. Modificar activo físico a nivel de proyecto

RF5.3. Eliminar activo físico a nivel de proyecto

RF5.4. Mostrar activo físico a nivel de proyecto

RF6. Filtrar activos a nivel de proyecto

RF6.1. Filtrar por categorías de activos registrados a nivel de proyecto

RF6.2. Filtrar por estados de activos registrados a nivel de proyecto

RF7. Calcular automáticamente el valor actual y la depreciación acumulada de todos los activos registrados en el sistema

RF8. Permitir la dependencia entre activos

RF8.1 Mostrar los activos dependientes

RF8.2 Eliminar activos dependientes

RF9. Buscar Activo

RF9.1. Buscar activo por código

RF9.2. Buscar activo por título

RF9.2. Buscar activo por descripción

2.4.2. Requisitos no funcionales.

Los RNF surgen de la necesidad del cliente debido a las restricciones en el presupuesto, las políticas de la organización, la necesidad de interoperabilidad con otros sistemas de software o hardware o a factores externos como los reglamentos de seguridad, las políticas de privacidad, entre otros. Los RNF forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto. Estos diferentes tipos de requerimientos se clasifican de acuerdo con sus implicaciones.

Tabla 2.1. Requisitos no funcionales

Hardware	
RNF1	Servidor Aplicaciones Web: 2GB RAM, 250 GB disco duro.
RNF2	PC cliente: prestaciones de hardware que permitan la instalación de cualquier Sistema Operativo, conexión de red.
Software	
RNF3	PC cliente: debe tener un navegador web, preferentemente Mozilla Firefox o Google Chrome.
RNF4	PC servidor de base de datos: El servidor debe contar con sistema operativo GNU/Linux Ubuntu 14.04 LTS y el SGDB PostgreSQL 9.4.
RNF5	Sistema de control de versiones: Git 1.9.1
RNF6	Como servidor de aplicación Apache 2.4
Restricciones de diseño e implementación	
RNF7	IDE de desarrollo: RubyMine 7.1.2.
RNF8	Paradigma de la arquitectura Modelo-Vista-Controlador (MVC).
RNF9	Lenguaje de programación: Ruby.
Eficiencia	
RNF11	El sistema debe soportar un tiempo de respuesta menor o igual a 5 segundos.
RNF12	El sistema debe soportar una conexión simultánea de más de 3000 usuarios.
Requisitos de apariencia o interfaz externa	
RNF13	La interfaz debe ser sencilla con colores suaves a la vista y sin cúmulo de imágenes u objetos que distraigan al cliente del objetivo.
RNF14	Debe utilizar como idioma principal el español, aunque debe existir traducción al inglés.
RNF15	Los botones deben expresar su función ya sea mediante su texto o la imagen que lo acompañe.
Interfaces de hardware	
RNF16	La comunicación entre el servidor de aplicaciones y la base de datos se lleva a través del protocolo de conexión TCP/IP.
RNF17	La comunicación entre el cliente y el servidor de aplicaciones se lleva a través del protocolo HTTPS.
Requisitos Legales, de Derecho de Autor y otros	
RNF18	El sistema debe ser sometido a un análisis legal por parte de los abogados y personal autorizado con vistas a declarar su autenticidad y evitar restricciones legales para su uso y comercialización; así mismo se debe proceder a una evaluación y certificación por parte del cliente del producto.

Tabla 2.2. Requisitos no funcionales

Seguridad	
RNF19	El sistema debe estar disponible las 24 horas, los 365 días del año, para garantizar la ejecución de las tareas en el momento requerido.
RNF20	Permite el acceso al sistema a través de una cuenta de usuario única que puede ser local o por autenticación con el LDAP.
RNF21	El acceso por las cuentas de administración a los servidores se realiza a través de una contraseña generada con algoritmo MD5. El listado de las claves de cada servidor se almacena en un archivo comprimido cifrado.
RNF22	Los usuarios creados en la Suite GESPRO tienen acceso (o permiso) limitado, pueden acceder a un conjunto de funcionalidades según el rol o perfil donde fue clasificado el usuario, que a su vez este rol fue clasificado en un nivel para la toma de decisiones determinado.
Usabilidad	
RNF23	Facilidad de uso por parte de los usuarios: el sistema debe presentar una interfaz amigable que permita la fácil interacción con el mismo y llegar de manera rápida y efectiva a la información buscada. Debe, además, ser una interfaz de manejo cómodo que posibilite a los usuarios sin experiencia una rápida adaptación.
RNF24	Especificación de la terminología utilizada: el sistema debe adaptarse al lenguaje y términos utilizados por los usuarios en la rama abordada con vista a una mayor comprensión por parte del cliente de la herramienta de trabajo.
RNF25	El sistema debe presentar una serie de menús tanto laterales como en barra de íconos flotantes que permitan el acceso rápido a la información por parte de los usuarios, lo que aprovecha las potencialidades de estas estructuras.
Portabilidad	
RNF26	El sistema debe permitir su uso en diferentes sistemas operativos, como Linux y Windows.

2.4.3. Historias de usuarios asociadas a los requisitos funcionales:

Seguidamente se muestran las historias de usuarios asociadas a los requisitos funcionales definidos previamente. El resto de las historias de usuarios se encuentran registradas en los anexos de la investigación (ver anexo B.2).

Tabla 2.3. Historia de usuario # 1

Historia de usuario	
Número: 1	**Nombre:** Crear activo intangible a nivel de centro
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 0.8	**Iteración asignada:** 1
Programador responsable: Ernesto Soler Calaña	
Descripción: Permite crear un nuevo activo intangible a nivel de centro e incluirlo en el sistema, luego de haber llenado los datos correspondientes. En caso de que se cree correctamente el activo, el sistema debe mostrar un mensaje indicando que la operación fue realizada satisfactoriamente.	
Observaciones: En caso que el activo no sea creado correctamente, el sistema debe mostrar un mensaje de error con la explicación del mismo. Si el usuario no introduce los campos necesarios para la correcta creación del activo intangible, el sistema debe mostrar un mensaje de error, indicando los campos que faltan por llenar.	

Tabla 2.4. Historia de usuario # 2

Historia de usuario	
Número: 2	**Nombre:** Modificar activo intangible intangible a nivel de centro
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 0.5	**Iteración asignada:** 1
Programador responsable: Ernesto Soler Calaña	
Descripción: Permite modificar un activo intangible registrado a nivel de centro en el sistema. Si el usuario modifica correctamente el activo, el sistema debe mostrar un mensaje indicando que la operación fue realizada satisfactoriamente.	
Observaciones: En caso que el activo no sea modificado, el sistema debe mostrar un mensaje de error con la explicación del mismo. Si el usuario no introduce los campos necesarios para la correcta modificación del activo intangible, el sistema debe mostrar un mensaje de error, indicando los campos que faltan por llenar.	

2.5. Arquitectura de la solución:

El MVC es una propuesta de diseño de software utilizada para implementar sistemas donde se requiere el uso de interfaces de usuario. Surge de la necesidad de crear software más robusto con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de

conceptos (ALVAREZ, 2014).

El MVC es un patrón de diseño de software verdaderamente probado que convierte una aplicación en un paquete modular fácil de mantener y mejora la rapidez del desarrollo. La separación de las tareas de la aplicación en modelos, vistas y controladores hace que dicha aplicación sea además muy ligera de entender. Las nuevas características se añaden fácilmente y agregar métodos o nuevas funcionalidades se hace muy sencillo. El diseño modular también permite a los desarrolladores y los diseñadores trabajar simultáneamente, incluyendo la capacidad de hacer prototipos rápidos (CAKEPHP, 2016).

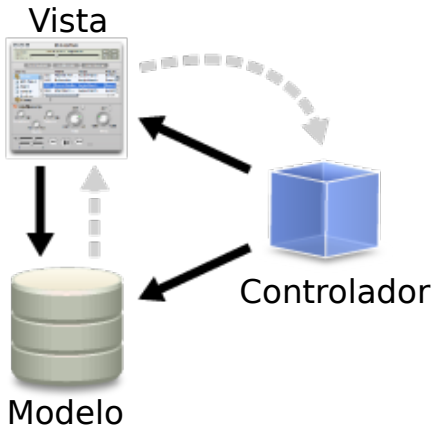


Figura 2.2. Patrón de arquitectura MVC

Esta arquitectura (ver 2.2) en RoR está estructurada de la siguiente forma (SAM RUBY, 2009):

Modelo: El modelo es responsable de mantener el estado de la aplicación. A veces, este estado es transitorio, es decir, dura sólo un par de interacciones con el usuario. En ocasiones, el estado es permanente y se almacena fuera de la aplicación, a menudo en una base de datos. Un modelo es algo más que datos; hace cumplir todas las reglas de negocio que se aplican a esos datos. El modelo actúa como un controlador de acceso y un almacén de datos.

Vista: La vista es responsable de generar una interfaz de usuario, basada normalmente, en los datos en el modelo. Aunque la vista puede presentar al usuario varias formas de introducir datos, la vista en sí nunca maneja los datos entrantes. El trabajo de la vista se realiza una vez que se muestran los datos.

Controlador: El controlador es el encargado de dirigir la aplicación. Los controladores reciben eventos "del mundo exterior" (normalmente la entrada del usuario), interactúan con el modelo, y visualiza una vista apropiada para el usuario.

El uso del patrón antes mencionado, sirve de apoyo para estructurar los componentes del módulo de manera tal que se obtenga una clara separación entre interfaz, lógica de negocio y presentación, brindando una mayor sencillez para crear distintas representaciones de los datos, además de proporcionar soporte a la reutilización de los componentes así como la simplicidad para un futuro mantenimiento.

2.5.1. Patrones de diseño

A continuación se relacionan los patrones de diseño empleados y su aplicación durante la implementación.

Patrones Generales de Software para Asignación de Responsabilidades (GRASP, por sus siglas en inglés): patrones que asignan responsabilidades a través de la descripción de los principios fundamentales del diseño y la solución a un problema. Entre estos se toman los siguientes patrones :

Experto: Se establece la asignación de responsabilidades específicas por cada una de las clases del sistema. Se cuenta con las clases controladoras, modelos y entidades, que poseen funcionalidades específicas atendiendo a la actividad que realizan y los procesos que gestionan.

Creador: Se aplica para la asignación de responsabilidades a las clases relacionadas con la creación de objetos, de forma tal que una instancia de un objeto sólo puede ser creada por el objeto que contiene la información necesaria para ello. El uso de este patrón permite crear las dependencias mínimas necesarias entre las clases, lo que favorece al mantenimiento del sistema y ofrece mejores oportunidades de reutilización. Se emplea este patrón para alcanzar también el bajo acoplamiento, encapsular y posibilitar la reutilización de código. Las clases controladoras son las encargadas de crear los modelos y estas a su vez las entidades garantizando con ello la distribución por capas de la arquitectura.

Controlador: Las clases Controladoras son las encargadas de gestionar el acceso a la lógica de negocio y controlar todo el proceso. En caso de que sean muy extensas, estas se dividen en varias controladoras para separar la carga de procesamiento, disminuir la complejidad y responsabilidad de las clases. Para ello se definieron controladores genéricos que contiene los métodos comunes para otras clases controladoras.

Bajo acoplamiento: Se diseña bajo este principio, lo que permite el desarrollo por separado a partir de las especializaciones individuales de cada uno. El intercambio de información se realiza a través de servicios implementados que se encargan de distribuir la información para la realización de procesos dependientes. Además, permitir la menor dependencia posible entre las clases de manera que si se realiza una modificación en alguna de ellas no afecte el funcionamiento de las otras. En la solución propuesta se evidencia mediante la independencia de la misma respecto a otros módulos registrados en la suite GESPRO.

Alta cohesión: El diseño y la dependencia entre clases están elaborados a partir de las funcionalidades que realizan, presentando alta relación de afinidad en las operaciones que controlan. En el caso de las actividades de alta complejidad comparten relación con otros objetos, disminuyendo la carga de transacciones entre ellas.

Polimorfismo: Cuando las alternativas o comportamientos relacionados varían según el tipo (clase), se tiene que asignar la responsabilidad del comportamiento, utilizando las operaciones polimorfas, a los tipos para los que varía el comportamiento. Según (ibíd.) : una asociación polimórfica es una asociación que une a los objetos de diferentes tipos. Supuestamente estos objetos comparten algunas características comunes, pero que van a tener diferentes representaciones. En el caso de la solución propuesta, se emplea para la relación entre los activos físicos e intangibles. Cada uno de los diferentes tipos de activos le corresponde una tabla de base de datos y un modelo diferente, pero todos ellos son activos.

2.5.2. Patrón de acceso a datos

El patrón de acceso a datos es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos. Active Record (AR) sigue el estándar de Mapeo Objeto-Relacional (ORM, por sus siglas en inglés) y se diferencia de los demás porque minimiza la cantidad de configuraciones mediante el uso de un conjunto de convenciones. Cada clase AR representa una tabla de la base de datos cuyos atributos son representados como las propiedades de la clase y una instancia AR representa una fila en esa tabla. Las operaciones CRUD comunes son implementadas como métodos de la clase. Como resultado, se puede acceder a los datos de una manera más orientada a objetos.

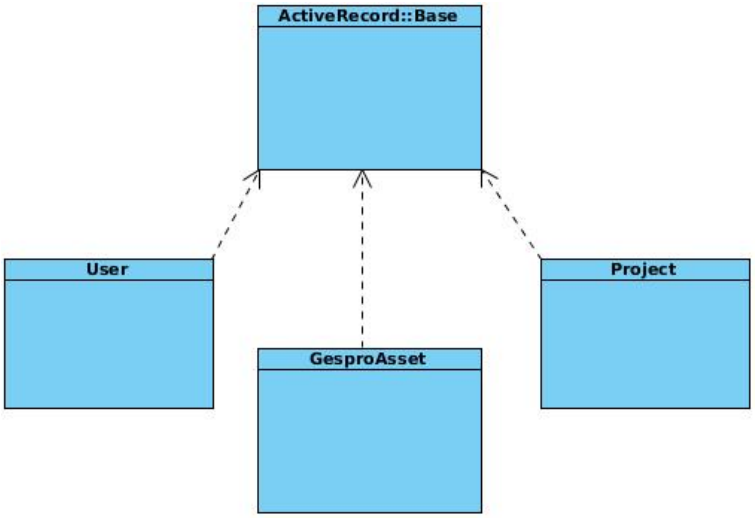


Figura 2.3. Ejemplo del uso del Patrón ActiveRecord

2.6. Diagrama de componentes

El diagrama de componentes permite visualizar con facilidad la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces. En ellos se muestran los elementos de diseño de un sistema de software y sus dependencias. En el diagrama de componentes del módulo de gestión de activos se agrupan los componentes comunes en paquetes, y se define la relación entre éstos (2.4). De esta manera, todos los componentes controladores están dentro del paquete *controller*, los componentes vistas están dentro del paquete *view* y los componentes modelos están dentro del paquete *model*. Además se representan componentes externos como las bases datos y clases auxiliares utilizadas.

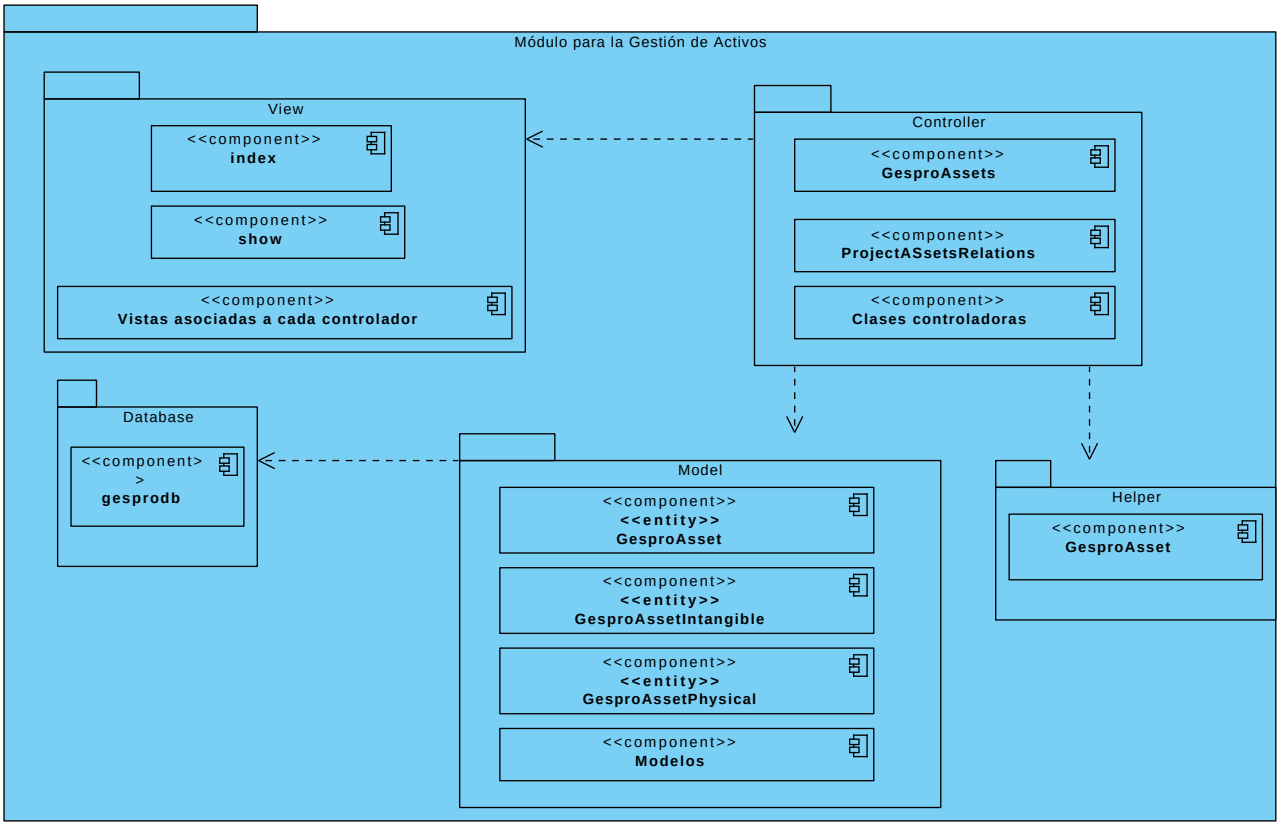


Figura 2.4. Diagrama de componentes

2.7. Modelo Entidad Relación

El Modelo de Entidad Relación (MER) es un modelo de datos basado en una percepción del mundo real que consiste en un conjunto de objetos básicos llamados entidades y relaciones entre estos objetos, implementándose en forma gráfica a través del Diagrama Entidad Relación (GUILLERMO STORTI, 2007).

Una entidad es un objeto que existe y puede distinguirse de otros objetos; entonces, un conjunto de entidades, es un grupo de entidades del mismo tipo (CIENCIAS AGRARIAS, 2015).

Con este modelo se logra representar de manera gráfica la estructura lógica de la base de datos. El módulo de gestión de activos incorpora tablas necesarias para dar respuesta a la propuesta de solución. La Figura 2.5 muestra la estructura de la base de datos del módulo de gestión de activos de GESPRO.

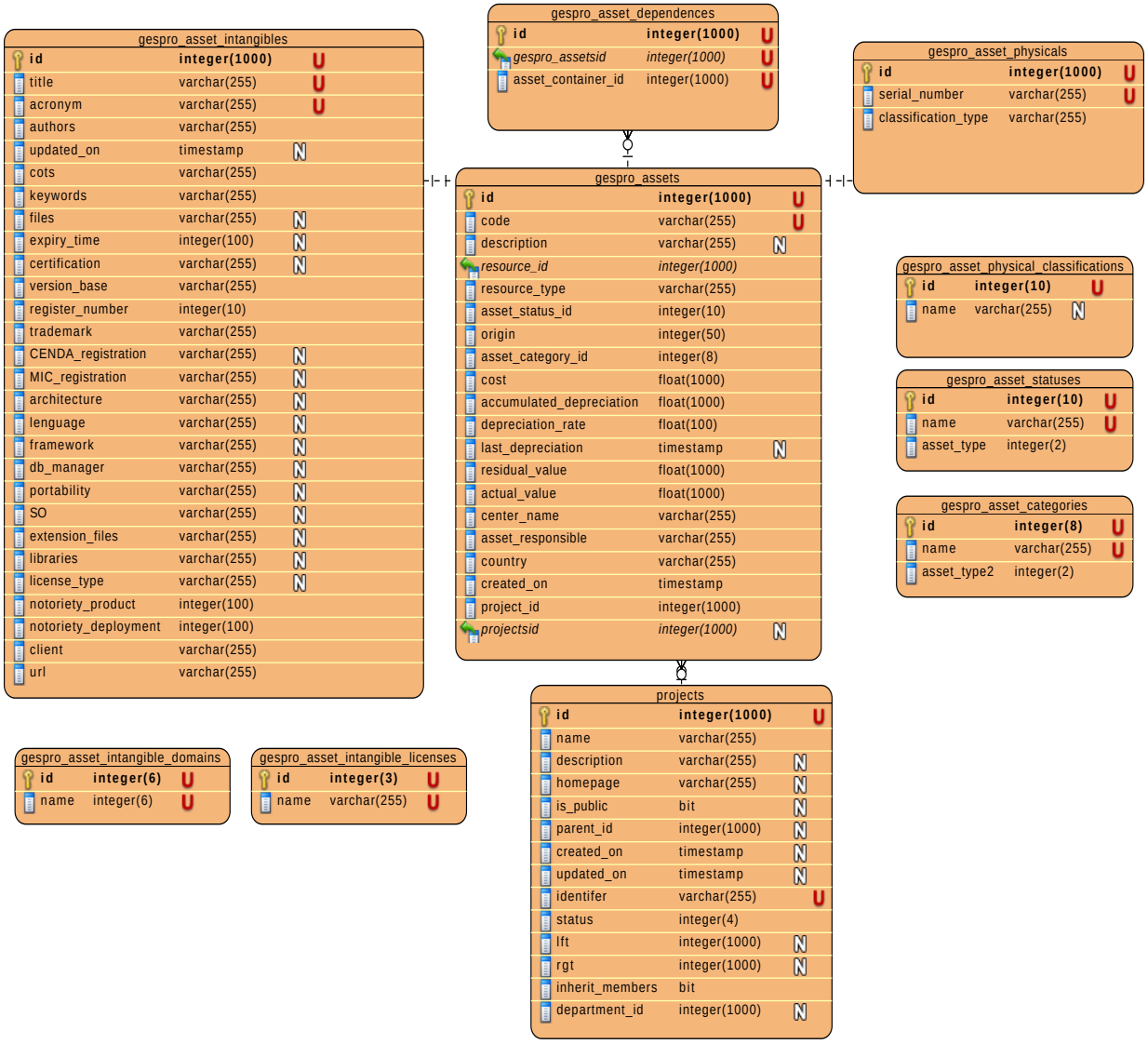


Figura 2.5. Diagrama Entidad Relación

gespro_assets: en esta tabla se almacenan todos los activos registrados en GESPRO, el atributo resource_id permite establecer una relación de uno a uno según el tipo de activo (resource_type) que se desea almacenar en la base de datos.

gespro_asset_intangible: en esta tabla se almacenan todos los activos de tipo intangible. Contiene una relación de uno a uno con la tabla gespro_assets.

gespro_asset_physical: en esta tabla se almacenan todos los activos de tipo físicos. Contiene una relación de uno a uno con la tabla gespro_assets.

gespro_asset_dependences: en esta tabla se almacena la relación de uno a muchos entre activos según su dependencia, es decir se registra si un activo es hijo o padre de otro.

project: en esta tabla se almacenan todos los proyectos creados en GESPRO con sus respectivos atributos.

gespro_asset_statuses: en esta tabla se almacenan los tipos de estados que pueden tener los activos. Constituye una tabla persistente en la que solo se pueden eliminar los datos de forma manual.

gespro_asset_categories: en esta tabla se almacenan las diferentes categorías que pueden presentar los activos. Constituye una tabla persistente en la que solo se pueden eliminar los datos de forma manual.

gespro_asset_intangible_domains: en esta tabla se almacenan los tipos de dominio que pueden tener los activos intangibles. Constituye una tabla persistente en la que solo se pueden eliminar los datos de forma manual.

gespro_asset_physical_classification: en esta tabla se almacenan los tipos de clasificaciones que pueden tener los activos físicos. Constituye una tabla persistente en la que solo se pueden eliminar los datos de forma manual.

gespro_asset_intangible_licenses: en esta tabla se almacenan los tipos de licencia que pueden tener los activos intangibles. Constituye una tabla persistente en la que solo se pueden eliminar los datos de forma manual.

2.8. Conclusiones parciales.

La descripción en lenguaje natural de las necesidades del cliente, posibilita la especificación de los requisitos, funcionales y no funcionales, identificando así las funciones que debe realizar el módulo y las restricciones que este debe cumplir. La utilización del patrón arquitectónico MVC permite estructurar los componentes del módulo, de esta manera, se obtiene una clara separación entre interfaz, lógica de negocio y datos, lo cual sirve de soporte a la reutilización de los componentes reduciendo el costo y tiempo de desarrollo. La definición de los patrones, tanto arquitectónico, de diseño y acceso a datos a utilizar permite establecer la base estructural de la implementación del módulo.

CAPÍTULO 3

Implementación y pruebas

3.1. Introducción

En el presente capítulo se realiza la implementación y validación de la solución con sus respectivos artefactos que se generan según la metodología previamente seleccionada. Se muestra el diagrama para el despliegue del sistema a implementar. Se determina la validez de la propuesta desarrollada mediante la realización de pruebas al módulo y los resultados que estas generan, con el objetivo de garantizar el correcto funcionamiento de los requisitos identificados del sistema a desarrollar.

3.2. Implementación del módulo

Dentro del ciclo de vida de un sistema informático se encuentra la fase de implementación, muy costosa en cuanto a tiempo y recursos ya que se ponen en práctica todas las descripciones y arquitecturas propuestas en la fase de análisis y diseño. La implementación es el complemento del trabajo de las fases anteriores dentro del proceso unificado de software en la que se completa el trabajo realizado previamente, se materializan los requisitos y se obtiene como resultado componentes de código que se compilan e integran en versiones ejecutables.

3.2.1. Estructura del módulo

El módulo para la gestión de activos se encuentra estructurado de acuerdo a la siguiente distribución (3.1):

1. **app**: contiene el núcleo del módulo dividido en tres subdirectorios: *controllers*, *models*, *views* y *helpers*.
2. **controllers**: contiene las clases controladoras del módulo.
3. **models**: contiene las clases de modelación de datos almacenados en la base de datos de la aplicación.

4. **views**: contiene las plantillas de visualización que se llenan con los datos de la aplicación, las mismas son convertidas en HTML y devueltas al navegador del usuario.
5. **helpers**: contiene las clases auxiliares que proporcionan funcionalidades complementarias.
6. **config**: contiene el subdirectorio *locales* y el archivo *routes.rb*.
   a) **locales**: se almacenan los archivos dedicados a los idiomas de la aplicación permitiendo la fácil migración a cualquiera de estos. El módulo brinda soporte para inglés y español.
   b) **routes**: se especifican las rutas para acceder a las interfaces.
7. **db**: contiene el subdirectorio *migrate*.
   a) **migrate**: contiene archivos SQL para la creación y modificación de tablas en la base de datos. Los migrates permiten trabajar independiente del servidor central, y una vez terminado, es posible migrar la base de datos central.

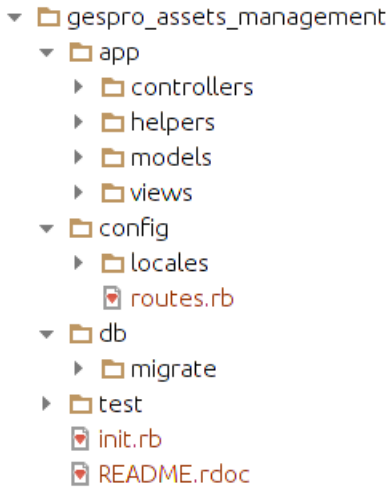


Figura 3.1. Directorio del módulo

3.2.2. Descripción de las clases

Clases Controladoras (app/controller)	
BaseAssetApplicationController	Clase padre de todas las clases controladoras. Contiene todos los métodos genéricos heredados por otras clases.
GesproAssetsController	Clase base para la gestión de activos, contiene métodos genericos para la los activos a nivel de proyecto.
ProjectAssetsRelationsController	Clase encargada de la asociación o dependencia entre activos a nivel de proyecto.
GesproAssetsIntangibleController	Clase encargada de la gestión de los activos intangibles a nivel de proyecto.
GesproAssetPhysicalController	Clase encargada de la gestión de los activos físicos a nivel de proyecto..
CenterGesproAssetsController	Clase base para la gestión de activos, contiene métodos genericos para la los activos a nivel de centro.
CenterAssetsRelationsController	Clase encargada de la asociación o dependencia entre activos a nivel de centro.
CenterAssetsIntangibleController	Clase encargada de la gestión de los activos intangibles a nivel de centro.
CenterAssetPhysicalController	Clase encargada de la gestión de los activos físicos a nivel de centro..

Figura 3.2. Descripción de las clases controladoras del módulo.

Clases Modelos (app/models)	
GesproAsset	Posibilita la conexión directa con la tabla gespro_assets, en la cual se almacenan los activos.
GesproAssetPhysical	Posibilita la conexión directa con la tabla gespro_asset_physicals, en la cual se almacenan todos los activos físicos.
GesproAssetIntangible	Posibilita la conexión directa con la tabla gespro_asset_intangibles.
GesproAssetCategory	Posibilita la conexión directa con la tabla gespro_asset_categories, en la cual se almacenan las categorias que podría tener un activo.
GesproAssetStatus	Posibilita la conexión directa con la tabla gespro_asset_status, en la cual se almacenan los tipos de estado que puede tener un activo.
GesproAssetDependence	Posibilita la conexión directa con la tabla gespro_asset_dependences, en la cual se almacenan las relaciones de uno a mucho entre los activos, mostrando la dependencia o asociación de los activos registrados.
GesproAssetIntangibleDomain	Posibilita la conexión directa con la tabla gespro_asset_intangible_domains, en la cual se almacenan los tipos de dominio que puede tener un activo intagible.
GesproAssetIntangibleLicense	Posibilita la conexión directa con la tabla gespro_asset_intangible_licenses, en la cual se almacen los tipos de licencia q puede tener un activo intangible.

Figura 3.3. Descripción de las clases modelos del módulo.

Clases auxiliares (.../...)	
GesproAssetsHelper	Provee métodos auxiliares para las funcionalidades asociadas a la gestión de activos.

Figura 3.4. Descripción de las clases auxiliares del módulo.

Las clases controladoras son las encargadas de controlar todo el proceso por el cual transcurre el negocio. Se encarga de la lógica de la petición, haciendo a su vez un puente entre el modelo y la vista. Cada controlador es una clase con métodos, y por cada método hay una vista que representa la versión procesada de ese método. El modelo está formado por clases, cada clase es un modelo, y cada modelo representa una tabla en la base de datos cuyos atributos son representados como las propiedades de la clase y una instancia representa una fila de dicha tabla. Siendo el modelo, el encargado de trabajar con la lógica de la base de datos. Para agregar estas tablas se utilizan los archivos *migrate (migración)*, los cuales son sentencias de Lenguaje de Definición de Datos (DDL, por sus siglas en inglés) escritas en Ruby, que permiten administrar el esquema de la BD. Cuando se ejecuta un *migrate* se cambia la versión de un esquema a otra anterior o posterior. Las migraciones son almacenadas en archivos en el directorio *dbmigrate*, uno por cada clase. El nombre del archivo es de la forma **YYYYMMDDHHMMSS_nombre_fichero.rb**, esto es la marca de tiempo Tiempo Universal Coordinado (UTC, por sus siglas en inglés) identificando la migración seguida por un '_' seguido del nombre de la migración. El nombre de clase de la migración debe coincidir con la última parte del nombre de archivo. Por ejemplo **20161101152512_create_gespro_assets.rb** debe ser definida como **CreateGesproAssets**.

Código fuente 3.1. Ejemplo de *migrate* para agregar la tabla gespro_assets a la base de datos.

```
class CreateGesproAssets < ActiveRecord::Migration
  def self.up
    create_table :gespro_assets do |t|
      t.column :code, :string, :null => false
      t.column :title, :string, :null => false
      t.column :description, :string
      t.column :resource_id, :integer
      t.column :resource_type, :string
      t.belongs_to :asset_status, :null => false
      t.column :origin, :string, :null => false
      t.belongs_to :asset_category, :null => false
      t.column :cost, :float, :null => false
      t.column :accumulated_depreciation, :float, :null => false
      t.column :depreciation_rate, :float, :null => false
      t.column :last_depreciation, :timestamp
      t.column :residual_value, :float, :null => false
      t.column :actual_value, :float, :null => false
      t.column :center_name, :string, :null => false
      t.column :asset_responsible, :string, :null => false
      t.column :country, :string, :null => false
```

```
      t.column :created_on, :timestamp, :null =>false
      t.belongs_to :project, :null => false

    end
  end

  def self.down
    drop_table :gespro_assets
  end
end
```

Este *migrate* crea una tabla llamada gespro_assets con 17 columnas que constituyen los atributos de la clase GesproAsset. Además de una columna llave primaria llamada *id* es creada por omisión.

3.2.3. Diagrama de despliegue

Los Diagramas de despliegue muestran las relaciones físicas entre los componentes hardware y software en un sistema final. Un diagrama está compuesto por nodos y el reparto de los componentes sobre dichos nodos.(CERDA, 2003)

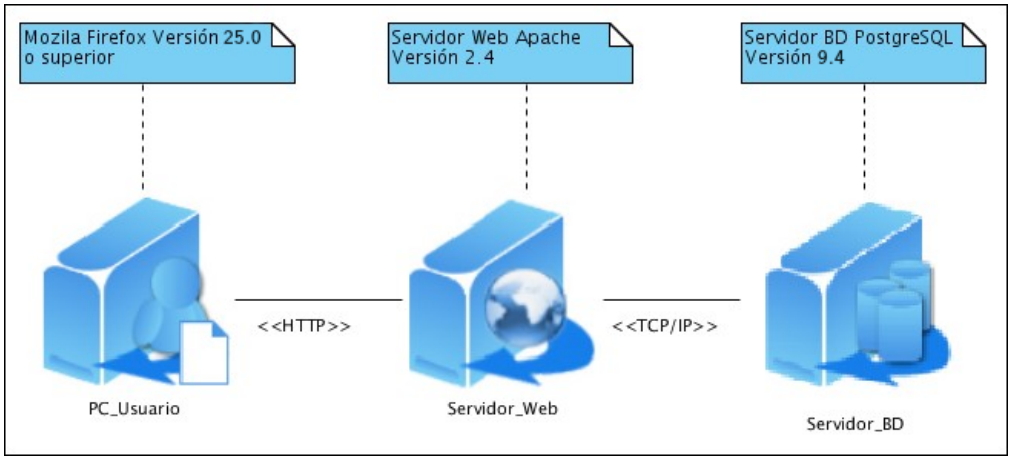


Figura 3.5. Diagrama de despliegue.

El nodo servidor Base de Datos (BD, por sus siglas en inglés) está conectado con el nodo servidor web mediante el protocolo de red TCP/IP, este a su vez está conectado con los nodos clientes a través del Protocolo Seguro de Transferencia de Hipertexto (HTTPS, por sus siglas en inglés), garantizando que en cada estación de trabajo los usuarios tengan acceso a la herramienta de manera segura. GESPRO está instalado en un servidor web Apache, el mismo, utiliza PostgreSQL como sistema gestor de base de datos.

3.3. Pruebas

Las pruebas al software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación. (R. S. PRESSMAN, 2002). Estas pruebas deben apoyarse en estándares que revisan los aspectos fundamentales que debe considerar todo proceso de pruebas. El proceso de pruebas se dirige fundamentalmente a componentes del software o al sistema de software en general, con el objetivo de intentar romper con la firmeza del software. Buscan hacer cumplir con las funcionalidades solicitadas por el cliente tratando de corregir todos los errores antes de entregar el programa al cliente. El principal objetivo del diseño de casos de prueba es obtener un conjunto de pruebas que tengan la mayor probabilidad de descubrir los defectos del software. (R. PRESSMAN, 2005)

3.3.1. Validación de las funcionalidades mediante las pruebas de caja negra

Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software. (R. S. PRESSMAN, 2002) Las pruebas funcionales de caja negra se refieren a las pruebas que se llevan a cabo sobre la interfaz del software, es por ello que, los casos de prueba pretenden constatar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como de garantizar que la integridad de la información externa se mantiene. Conociendo las funciones específicas para la que fue diseñado el módulo, se puede definir un conjunto de pruebas que demuestren la calidad y efectividad de dichas funcionalidades. Estas pruebas se realizan sobre las interfaces del módulo, declarando varias entradas de datos para luego estudiar las salidas que permiten corroborar que estas son las esperadas. Seguidamente se muestran los resultados obtenidos a partir de la aplicación de pruebas funcionales de caja negra a una de las funcionalidades que cuenta el módulo. El resto de las pruebas aplicadas se encuentran en la carpeta de anexos de la investigación (ver anexos C.1).

- **Escenario 1: Crear un activo a nivel de proyecto:** En la figura 3.6 que aparece a continuación se muestra el diseño de caso de prueba genérico para crear un activo intangible o físico, llenando los campos comunes entre ellos.Seguidamente en la tabla 3.1 se realiza la descripción de las variables necesarias para ejecutar este caso de prueba. Es necesario tener en cuenta que para crear un activo se deben llenar los campos correspondientes del tipo de activo que se elija, es decir, activo físico o intangible, ya que los campos mostrados en la tabla 3.1, son los genéricos para estos típos de activos, por lo que para crear un activo se necesitan llenar los campos genéricos mostrados a continuación más los campos requeridos del tipo de activo en cuestión. Posteriormente, se muestran las pruebas realizadas para cada tipo de activo*(intangibles y físicos)*.

Esc.	Descrip.	Código	Título	Estado	Origen	Categ.	Costo	T.Depre.	Valor Residual.	Centro	Reps.	País	R.S
E.C. 1 Crear un activo a nivel de proyecto de manera correcta.	El sistema permite crear un nuevo activo a nivel de proyecto llenando los campos genéricos.	V Ingresar el código.	V Ingresar el título.	V Ingresar el estado.	V Ingresar el origen.	V Ingresar la categoría.	V Ingresar el costo.	V Ingresar Tasa de Depreciación.	V Ingresar Valor Residual.	V Ingresar el centro.	V Ingresar el responsable.	V Ingresar el país del activo.	Se muestra mensaje de éxito. Redireccion a a la vista principal.
E.C.2 Crear un activo a nivel de proyecto de manera incorrecta.	El sistema no permite crear un nuevo activo a nivel de proyecto porque alguno/s de los campos no existen o no tienen el formato correspondiente	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	Se muestra un mensaje indicando los campos vacíos o con errores. Se mantiene en la vista.
		V Ingresar el código	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	
		I (vacío)	V Ingresar el título	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	
		I (vacío)	I (vacío)	V Ingresar el estado	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	
		I (vacío)	I (vacío)	I (vacío)	V Ingresar el origen	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	
		I (vacío)	I (vacío)	I (vacío)	I (vacío)	V Ingresar la categoría.	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	
		I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	V Ingresar el costo.	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	
		I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	V Ingresar Tasa de Depre.	I (vacío)	I (vacío)	I (vacío)	I (vacío)	
		I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	V Ingresar Valor Residual.	I (vacío)	I (vacío)	I (vacío)	
		I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	V Ingresar el centro del activo	I (vacío)	I (vacío)	
		I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	V Ingresar el responsable	I (vacío)	
		I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	V Ingresar el país	

Figura 3.6. Diseño del caso de prueba crear un activo (campos genéricos)

Tabla 3.1. Descripción de variables (3.6)

No.	Nombre del campo	Clasificación	Valor nulo	Descripción
1	Código	Campo de texto	No	Código del activo
2	Título	Campo de texto	No	Título del activo
3	Estado	Lista desplegable	No	Estado del activo
4	Origen	Lista desplegable	No	Centro de origen del activo
5	Categoría	Lista desplegable	No	Categoría del activo
6	Costo	Campo de texto	No	Costo del activo
7	Tasa de depreciación /amortización	Campo de texto	No	Tasa de depreciación/amortización del activo
8	Valor residual	Campo de texto	No	Valor residual del activo
9	Nombre del centro	Lista desplegable	No	Nombre del centro donde se encuentra
10	Responsable	Lista desplegable	No	Responsable del activo
11	País	Lista desplegable	No	País de confección u origen

Funcionalidad #4: Gestionar activos intangibles a nivel de proyecto

- **Escenario 1.2: Crear un activo intangible a nivel de proyecto de forma correcta:**
  La figura 3.7 muestra el diseño de caso de pruebas para los activos intangibles a nivel de proyecto, los cuales poseen varios atributos particulares (3.2) .

Esc.	Descrip.	Siglas	Autores	P.C	Versión	V.B	N.R	M/N.C	N.Prod.	N.Desp.	Clientes	URL	R.S
E.C. 1.2 Crear un activo intangible a nivel de proyecto de manera correcta.	El sistema permite crear un nuevo activo intangible a nivel de proyecto llenando los campos genéricos.	V Ingresar las siglas del activo.	Ingresar los autores del activo	V Ingresar las palabras claves del activo.	V Ingresar la versión del activo.	V Ingresar la versión base del activo.	V Ingresar el N.R del activo.	V Ingresar M/N.C del activo.	V Ingresar la N.Prod. del activo.	V Ingresar la N.Desp. del activo.	Ingresar los clientes.	V Ingresar la URL del activo.	Se muestra mensaje de éxito. Redireccion a a la vista principal.
E.C.1.3Crear un activo intangible a nivel de proyecto de manera incorrecta.	El sistema no permite crear un nuevo activo intangible a nivel de proyecto porque alguno/s de los campos no existen o no tienen el formato correspondiente	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	Se muestra un mensaje indicando los campos vacíos o con errores. Se mantiene en la vista.
		V Ingresar las siglas del activo.	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	
		I (vacío)	V I Ingresar los autores del activo	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	
		I (vacío)	I (vacío)	V Ingresar las palabras claves del activo.	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	
		I	I	I	V	I	I	I	I	I	I	I	
		(vacío)	(vacío)	(vacío)	Ingresar la versión del activo.	(vacío)	(vacío)	(vacío)	(vacío)	(vacío)	(vacío)	(vacío)	
		I (vacío)	I (vacío)	I (vacío)	I (vacío)	V Ingresar la versión base del activo.	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	
		I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	V Ingresar el N.R del activo.	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	
		I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	V Ingresar M/N.C del activo.	I (vacío)	I (vacío)	I (vacío)	I (vacío)	
		I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	V Ingresar la N.Prod. del activo.	I (vacío)	I (vacío)	I (vacío)	
		I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	V Ingresar la N.Desp. del activo.	I (vacío)	I (vacío)	
		I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	V Ingresar los	I (vacío)	

Figura 3.7. Diseño del caso de prueba crear un activo intangible a nivel de proyecto

Tabla 3.2. Descripción de variables (3.7)

No.	Nombre del campo	Clasificación	Valor nulo	Descripción
1	Siglas	Campo de texto	No	Siglas del activo intangible
2	Autores	Campo de texto	No	Autores del activo intangible
3	Palabras claves	Campo de texto	No	Palabras claves del activo intangible
4	Versión	Campo de texto	No	Versión del activo intangible
5	Versión base	Campo de texto	No	Versión base del activo intangible
6	Número de registro	Campo de texto	No	Número de registro del activo intangible
7	Marca/Nombre corto	Campo de texto	No	Número de registro del activo intangible
8	Notoriedad de los productos	Campo de texto	No	Notoriedad de los productos del activo intangible
9	Notoriedad de despliegues	Campo de texto	No	Notoriedad de despliegues del activo intangible
10	Clientes	Campo de texto	No	Clientes del activo intangible
11	URL	Campo de texto	No	Dirección web del activo intangible

Funcionalidad #2: Gestionar activos físicos a nivel de centro

- **Escenario 1.4: Crear un activo físico a nivel de proyecto de forma correcta:**
  La figura 3.8 muestra el diseño de caso de pruebas para los activos físicos a nivel de proyecto, los cuales poseen atributos particulares (3.3). En las figuras 3.9, 3.10 y 3.11 se visualizan las interfaces que muestran el flujo del proceso para crear un activo físico.

Escenario	Descripción	Número de Serie	Respuesta del Sistema
E.C. 1.4 Crear un activo físico a nivel de proyecto de manera correcta.	V El sistema permite crear un nuevo activo físico a nivel de proyecto llenando los campos genéricos.	V Ingresar el número de serie del activo físico	Se muestra mensaje de éxito. Redirecciona a la vista principal.
E.C.1.5 Crear un activo físico a nivel de proyecto de manera incorrecta.	El sistema no permite crear un nuevo activo físico a nivel de proyecto porque alguno/s de los campos no existen o no tienen el formato correspondiente	I (vacío)	Se muestra un mensaje indicando los campos vacíos o con errores. Se mantiene en la vista.

Figura 3.8. Diseño del caso de prueba crear un activo físico a nivel de proyecto

Tabla 3.3. Descripción de variables (3.8)

No.	Nombre del campo	Clasificación	Valor nulo	Descripción
1	Número de serie	Campo de texto	No	Número de serie del activo físico

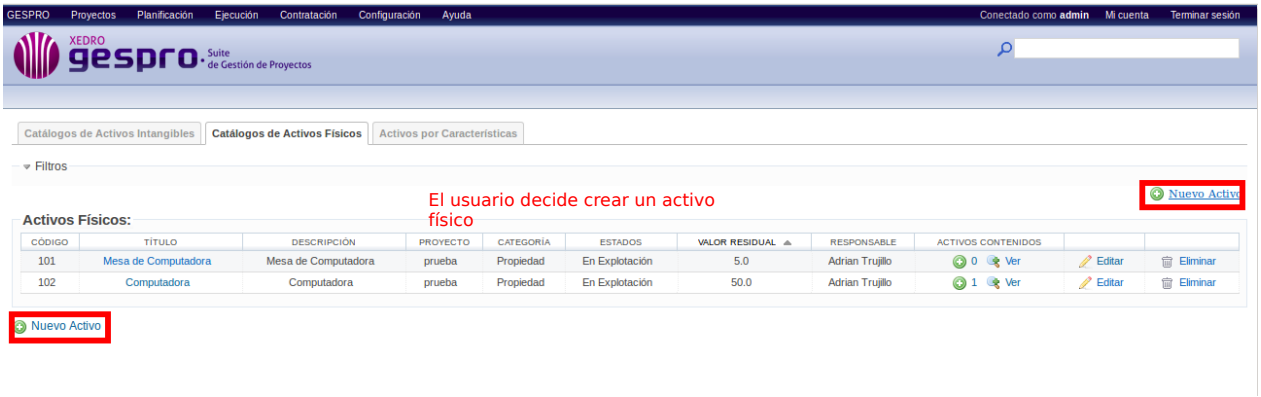


Figura 3.9. Crear un activo físico

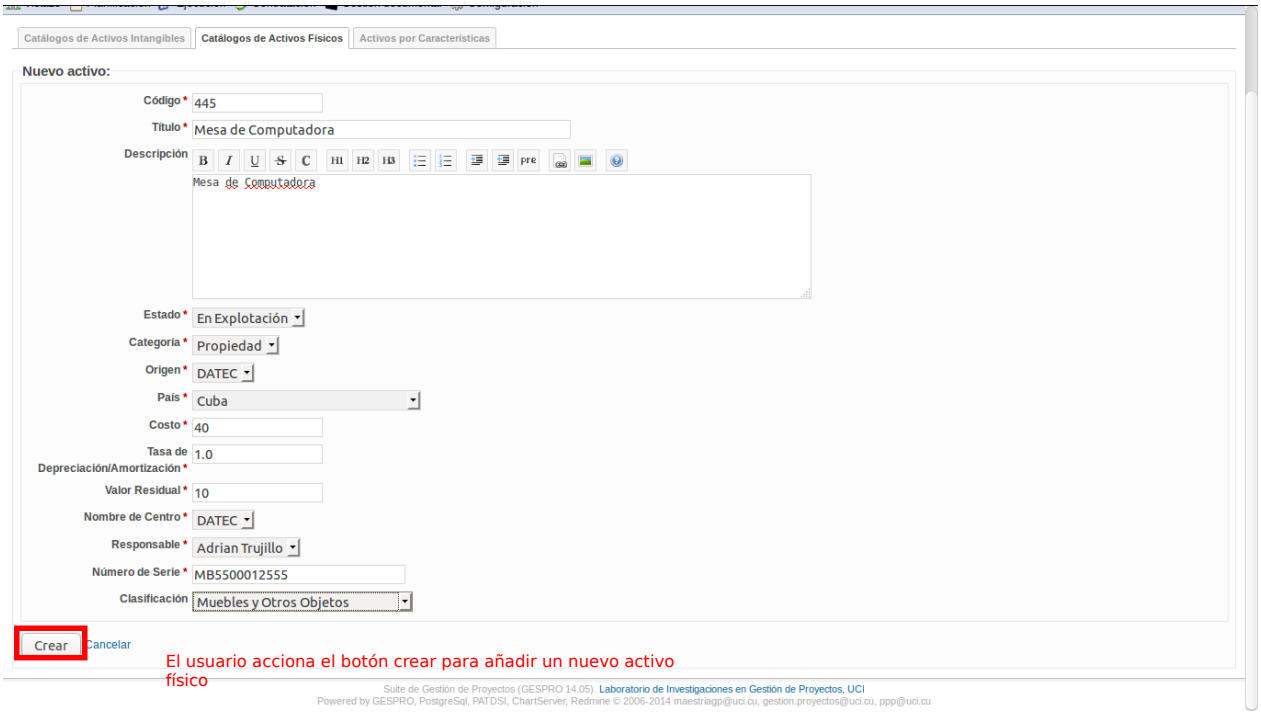


Figura 3.10. Formulario para crear un activo físico

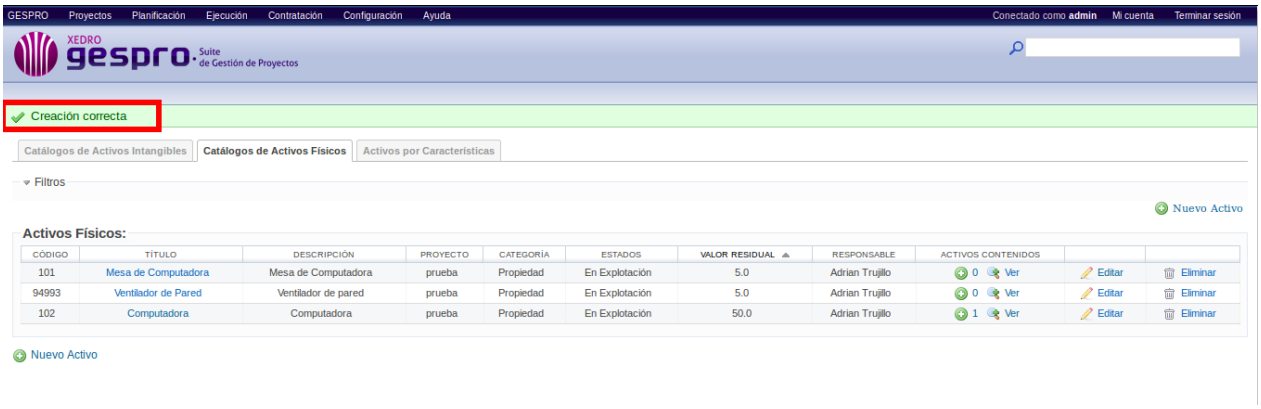


Figura 3.11. Respuesta del sistema al crear un activo físico correctamente

- **Escenario 1.5 Crear un activo físico de manera incorrecta:**
  La figura 3.12 muestra la respuesta del sistema a la acción de crear un activo físico sin introducir los datos correspondientes en los campos requeridos.

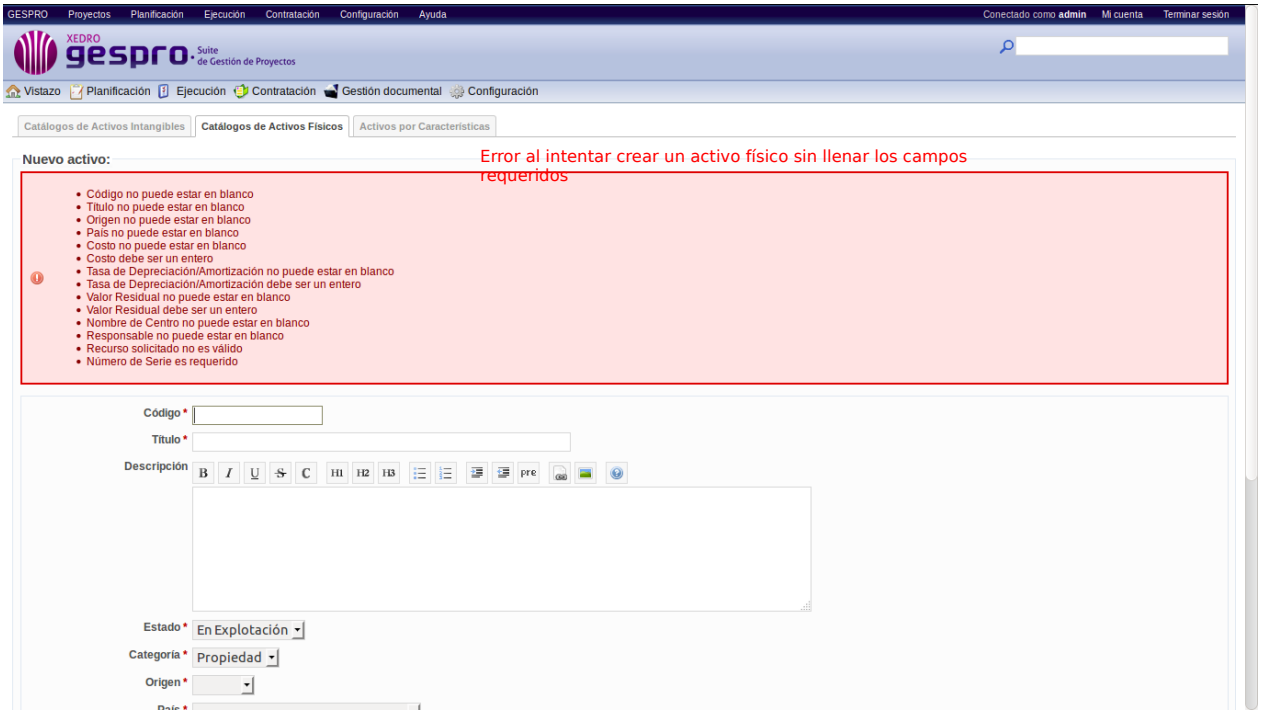


Figura 3.12. Respuesta del sistema al intentar crear un activo físico sin llenar los campos requeridos

- **Escenario 3. Modificar un activo a nivel de proyecto de manera correcta:**
  En la figura 3.13 se muestra el diseño del caso de prueba para modificar un activo a nivel de proyecto teniendo en cuenta los campos genéricos. Seguidamente, la tabla 3.4, muestra la descripción de las variables necesarias para ejecutar este caso de prueba.

Esc.	Descrip.	Código	Título	Estado	Origen	Categ.	Costo	T.Depre.	Valor Residual.	Centro	Reps.	País	R.S
E.C. 1.6 Modificar un activo a nivel de proyecto de manera correcta.	El sistema permite modificar un activo a nivel de proyecto llenando los campos genéricos.	V Ingresar el código.	V Ingresar el título.	V Ingresar el estado.	V Ingresar el origen.	V Ingresar la categoría.	V Ingresar el costo.	V Ingresar Tasa de Depreciación.	V Ingresar Valor Residual.	V Ingresar el centro.	V Ingresar el responsable.	V Ingresar el país del activo.	Se muestra mensaje de éxito. Redireccion a a la vista principal.
E.C.1.7 Modificar un activo a nivel de proyecto de manera incorrecta	El sistema no permite modificar activo a nivel de proyecto porque alguno/s de los campos no existen o no tienen el formato correspondiente	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	Se muestra un mensaje indicando los campos vacíos o con errores. Se mantiene en la vista.
		V Ingresar el código	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	
		I (vacío)	V Ingresar el título	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	
		I (vacío)	I (vacío)	V Ingresar el estado	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	
		I (vacío)	I (vacío)	I (vacío)	V Ingresar el origen	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	
		I (vacío)	I (vacío)	I (vacío)	I (vacío)	V Ingresar la categoría.	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	
		I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	V Ingresar el costo.	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	
		I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	V Ingresar Tasa de Depre.	I (vacío)	I (vacío)	I (vacío)	I (vacío)	
		I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	V Ingresar Valor Residual.	I (vacío)	I (vacío)	I (vacío)	
		I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	V Ingresar el centro del activo	I (vacío)	I (vacío)	
		I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	V Ingresar el responsable	I (vacío)	
		I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	V Ingresar el país	

Figura 3.13. Diseño del caso de prueba modificar un activo (campos genéricos) a nivel de proyecto

Tabla 3.4. Descripción de variables (3.13)

No.	Nombre del Campo	Clasificación	Valor Nulo	Descripción
1	Código	Campo de texto	No	Código del activo
2	Título	Campo de texto	No	Título del activo
3	Estado	Lista desplegable	No	Estado del activo
4	Origen	Lista desplegable	No	Centro de origen del activo
5	Categoría	Lista desplegable	No	Categoría del activo
6	Costo	Campo de texto	No	Costo del activo
7	Tasa de depreciación /amortización	Campo de texto	No	Tasa de depreciación/amortización del activo
8	Valor residual	Campo de texto	No	Valor residual del activo
9	Nombre del centro	Lista desplegable	No	Nombre del centro donde se encuentra
10	Responsable	Lista desplegable	No	Responsable del activo
11	País	Lista desplegable	No	País de confección u origen

- **Escenario 1.10. Modificar un activo físico a nivel de proyecto de manera correcta:**
  En la figura 3.14 se muestra el diseño del caso de prueba para la funcionalidad 5.2 modificar un activo físico a nivel de proyecto .La tabla 3.5, muestra la descripción de las variables necesarias para ejecutar este caso de prueba y posteriormente se visualizan las interfaces (3.15, 3.16, 3.17) que muestran el flujo del proceso para modificar un activo físico .

Escenario	Descripción	Número de Serie	Respuesta del Sistema
E.C. 1.10 Modificar un activo físico a nivel de proyecto de manera correcta.	V El sistema permite modificar un activo físico a nivel de proyecto llenando los campos requeridos.	V Ingresar el número de serie del activo físico	Se muestra mensaje de éxito. Redirecciona a la vista principal.
E.C. 1.11 Modificar un activo físico a nivel de proyecto de manera incorrecta.	El sistema no permite modificar un activo físico a nivel de proyecto porque alguno/s de los campos no existen o no tienen el formato correspondiente	I (vacío)	Se muestra un mensaje indicando los campos vacíos o con errores. Se mantiene en la vista.

Figura 3.14. Diseño del caso de prueba modificar un activo físico a nivel de proyecto

Tabla 3.5. Descripción de variables (3.14)

No.	Nombre del Campo	Clasificación	Valor Nulo	Descripción
1	Número de Serie	Campo de Texto	No	Número de Serie del Activo Físico

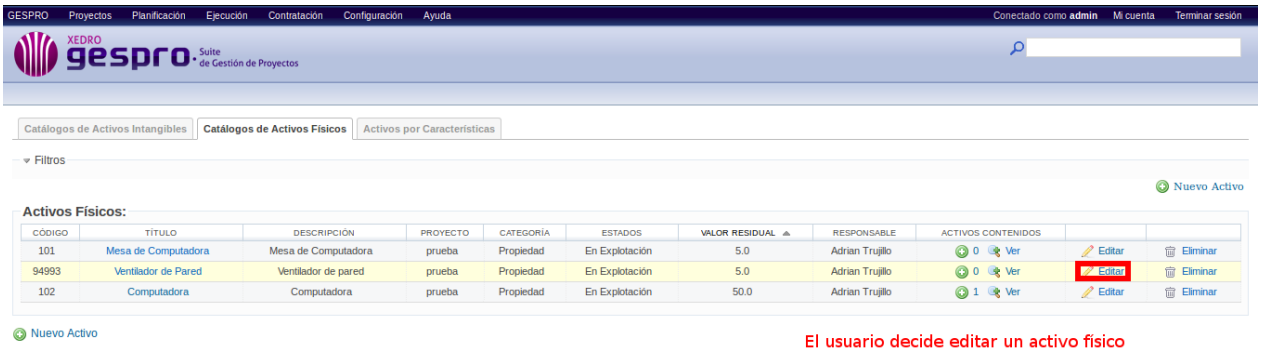


Figura 3.15. Modificar un activo

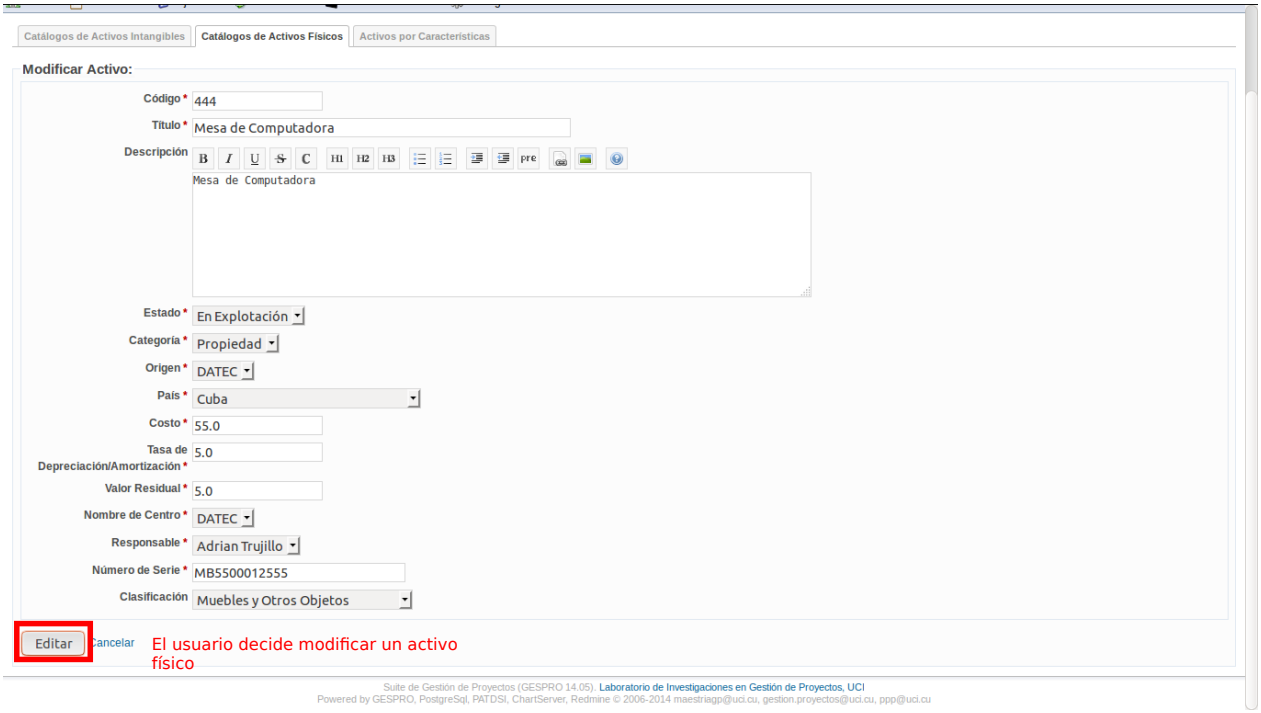


Figura 3.16. Formulario para modificar un activo físico

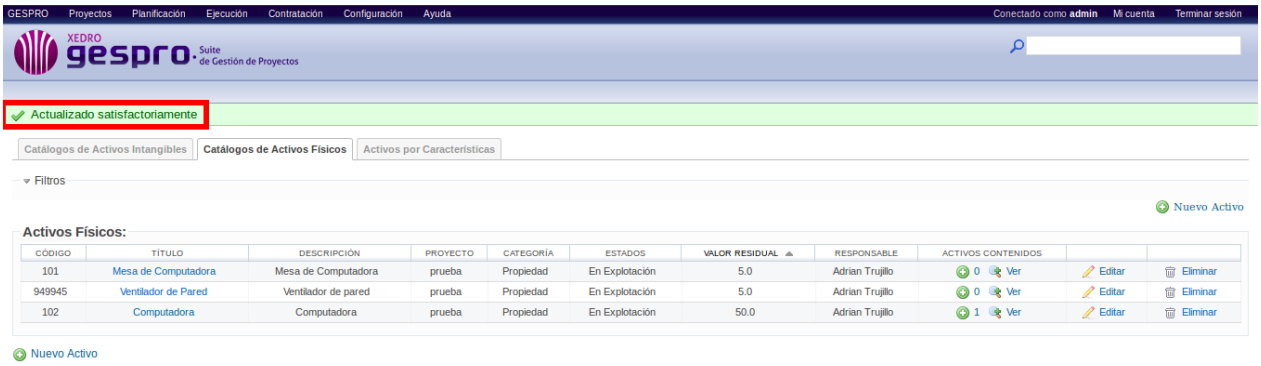


Figura 3.17. Respuesta del sistema al editar satisfactoriamente un activo físico

- **Escenario 5: Eliminar correctamente un activo**
  En la figura 3.18 se muestra el diseño de caso de prueba eliminar activo y en las figuras 3.19, 3.20 y 3.21 se visualizan las interfaces que muestran el flujo del proceso para eliminar un activo correctamente.

Escenario	Descripción	Confirmación del Sistema	Respuesta del Sistema
E.C. 1.12 Eliminar un activo a nivel de proyecto de manera correcta.	V El sistema permite eliminar un activo de forma correcta	V El sistema muestra un cartel para confirmar la eliminación del activo.	Se muestra mensaje de éxito. Redirecciona a la vista principal.

Figura 3.18. Diseño del caso de prueba eliminar un activo a nivel de proyecto

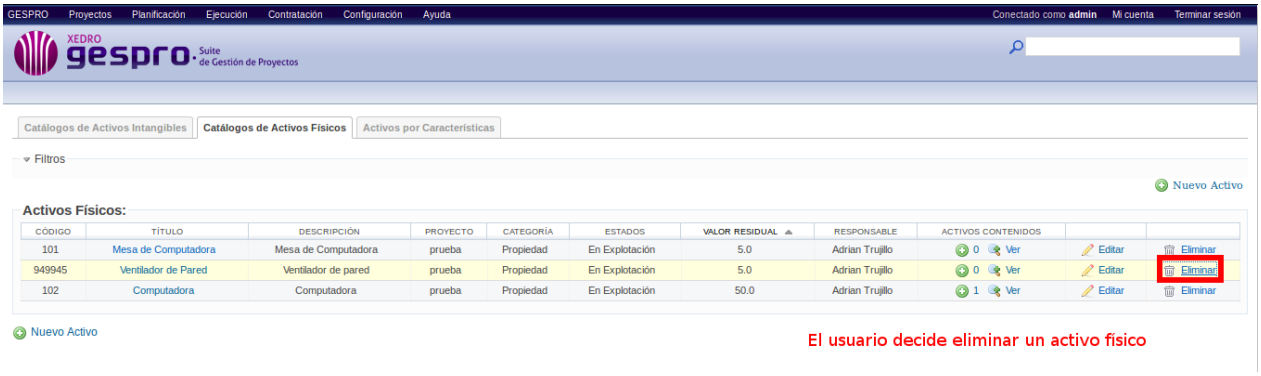


Figura 3.19. Eliminar un activo

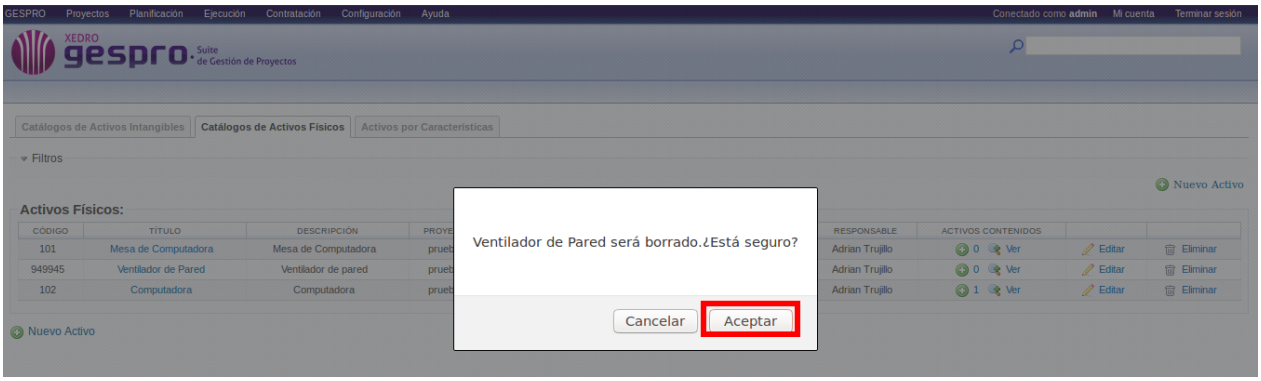


Figura 3.20. Mensaje de confirmación para eliminar un activo

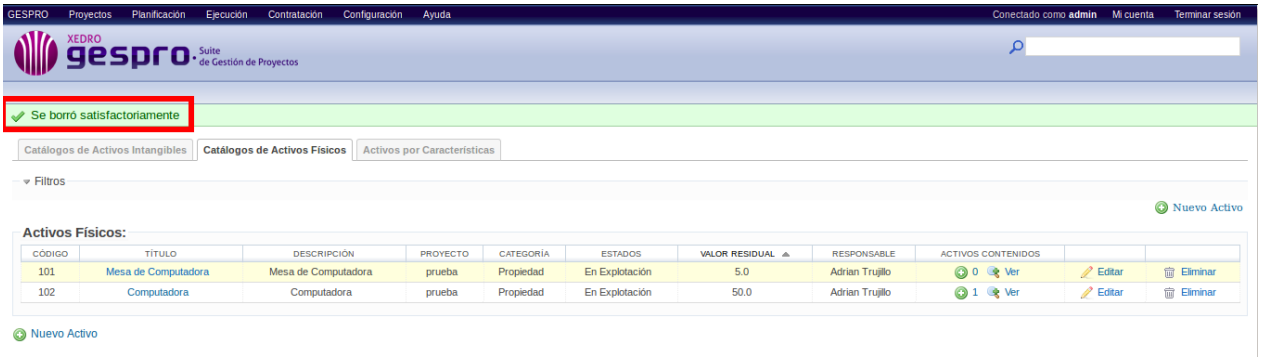


Figura 3.21. Respuesta del Sistema al eliminar un activo

3.3.2. Resultados obtenidos

Luego de realizar las pruebas pertinentes para probar el correcto comportamiento de las funcionalidades asociadas al módulo, se realizan 4 iteraciones(Ver 3.22) para detectar posibles no conformidades en la aplicación. Con las iteraciones realizadas se detectaron una serie de no conformidades que a lo largo de las pruebas han sido erradicadas. En la primera iteración se detectaron 6 no conformidades, en la segunda iteración 4 no conformidades, en la tercera iteración fueron detectadas 2 no conformidades y en la cuarta fueron erradicadas todas las no conformidades.

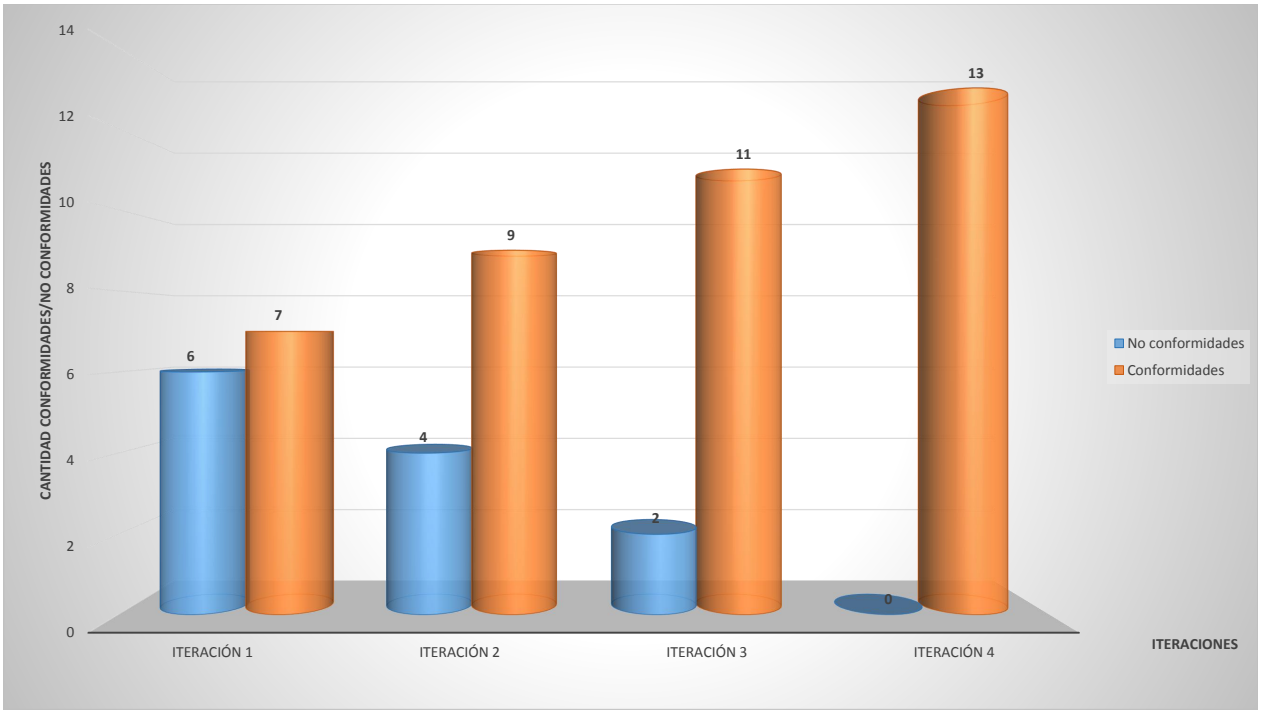


Figura 3.22. Gráfica 1: Resultados de las pruebas

3.4. Comparación con el módulo de Alcance Trazabilidad de la version 14.05 de GESPRO

La tabla 3.6 muestra un estudio comparativo de las principales funcionalidades que ofrecen los módulos para la gestión de activos de GESPRO. Con el objetivo de demostrar que la solución propuesta generaliza y a su vez se centra en los elementos fundamentales que necesita el módulo para una correcta gestión de todos los activos de la organización.

Tabla 3.6. Comparación del módulo Alcance Trazabilidad de GESPRO 14.05 y la propuesta de solución

Módulo	Gestión de activos intangibles	Gestión de activos físicos	Relación con requisistos	Depreciación	Dependencia entre activos
Módulo Alcance Trazabilidad	Si	No	Si	No	Si
Módulo de Activos	Si	Si	No	Si	Si

3.5. Impacto de la propuesta en GESPRO 16.05

Con el diseño e implementación del módulo para la gestión de activos para la suite GESPRO 16.05, se garantiza un eficaz control y gestión de los activos en centros y proyectos asociados a la suite que utilicen el módulo, potenciando a su vez la reutilización de activos de la organización. Constituye además una herramienta novedosa ya que abarca los activos físicos e intangibles, proporcionando la obtención de los valores de dichos activos en una organización orientada a proyectos.

En la UCI, el módulo para la gestión de activos para la suite GESPRO 16.05 mediante el uso de la solución propuesta, se pueden definir los proyectos y productos de software generados en los diferentes centros de producción de la entidad como activos intangibles, posibilitando la obtención del valor asociado a los mismos, su definición como útiles y a su vez la posibilidad de incluirlos como activos fijos potenciales de la organización.

En Cuba, la gestión de los activos constituye un reto, ya que muchas de las organizaciones orientadas a proyectos, desconocen cuan generalizados, relacionados y usados pueden ser los conceptos de **activos** y **reutilización**, en el control de sus activos fijos y a la hora de economizar gastos. La reutilización, por su parte es una actividad necesaria dada la situación económica en la que se encuentra el país, por lo que en aras de mejorar los servicios de diferentes organizaciones orientadas a proyectos, se evidencia la necesidad de herramientas que permitan una gestión generalizada de los activos de dichas organizaciones.

3.6. Conclusiones parciales

En este capítulo se detallan las tareas de la fase de implementación, obteniendo como resultado un módulo funcional para la gestión de activos proporcionando funcionalidades asociadas a las condiciones y necesidades actuales en esta área. Las descripciones de las clases proporcionan un mayor entendimiento

de las responsabilidades de las mismas al realizar sus respectivas funciones. La realización de las pruebas funcionales de caja negra permiten la corrección de errores, garantizando la validación de la implementación de las funcionalidades del módulo obteniendo resultados positivos para el problema de la investigación.

Conclusiones

- El análisis realizado a la solución existente en GESPRO 14.05 evidencia que la misma presenta limitantes las cuales impiden una correcta gestión de activos por lo que resulta necesario la realización del módulo para la gestión de activos de GESPRO 16.05.
- La solución propuesta garantiza la reutilización y el aprovechamiento de los activos (útiles) de una organización, tanto físicos como intangibles.
- El módulo propuesto posibilita la obtención de un valor para todos los activos registrados en la suite GESPRO.

Recomendaciones

- Integrar el módulo para la gestión de activos propuesto con el módulo de recursos, con el objetivo de relacionar los recursos registrados en el FCR con los activos registrados en la suite GESPRO.
- Definir grupos genéricos para los activos registrados en el módulo propuesto con el objetivo de replantear la forma de calcular la depreciación/amortización acumulada y valor actual de los mismos por una frecuencia a depreciar estándar para cada grupo.
- Mejorar el algoritmo de asociar activos optimizando el tiempo de respuesta del mismo.
- Por el diseño del módulo propuesto, puede ser empleado en cualquier tipo de proyecto.

Glosario

activos fijos bien de una empresa, ya sea tangible o intangible, que no puede convertirse en líquido a corto plazo y que normalmente son necesarios para el funcionamiento de la empresa y no se destinan a la venta.. 2, 53

Costo costo de producción o de compra, asociado a activo. 10

Crontab Programa que permite a usuarios Linux/Unix ejecutar automáticamente comandos o scripts (grupos de comandos) a una hora o fecha específica. Usado normalmente para comandos de tareas administrativas como respaldos.. 21

LDAP Protocolo compacto de acceso a directorios es un protocolo estándar que permite administrar directorios, esto es, acceder a bases de información de usuarios de una red mediante protocolos TCP/IP.. 26

Acrónimos

AR Active Record. 30

CASE Ingeniería de Software Asistida por Computadora. 17
CRUD crear, obtener, actualizar, eliminar. 12, 30
CSS Hoja de estilo en cascada. 18

DDL Lenguaje de Definición de Datos. 38
DIP Dirección Integrada por Proyectos. 1

FCR Fondo de Recursos Compartidos. 1, 2, 6

GPL Licencia Pública General. 23
GRASP Patrones Generales de Software para Asignación de Responsabilidades. 29
GUI Constructor de Interfaz Gráfica. 17

HTML Lenguaje de Marcas de Hipertexto. 17, 35
HTTPS Protocolo Seguro de Transferencia de Hipertexto. 39

IDE Entorno de Desarrollo Integrado. 17, 18

LGPL Licencia Pública General Reducida. 23

MER Modelo de Entidad Relación. 31
. 25, 27, 28, 33

NIC Normas Internacionales de Contabilidad. 9
NIIF Normas Internacionales de Información Financiera. 9

ORM Mapeo Objeto-Relacional. 30

RAE Real Academia de la Lengua Española. 7, 9
RF Requisitos funcionales. 23
RNF Requisitos no funcionales. 23, 24

SGBD Sistema de Gestión de Bases de Datos. 18

SGBDOO Sistema de Gestión de Bases de Datos Orientado a Objetos. 18

TIC Tecnologías de la Información y la Comunicación. 22

UCI Universidad de las Ciencias Informáticas. 1, 2, 53

UML Lenguaje de Modelado Unificado. 15, 17

UTC Tiempo Universal Coordinado. 38

Referencias bibliográficas

AJENJO, Alberto Domingo. 2005. *Dirección y Gestión de Proyectos. Un enfoque práctico*. Segunda Edición, 2005.

ALEGSA. 2016. *DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA*. 2016. Dirección: ⟨http://www.alegsa.com.ar/Dic/modulo.php⟩.

ALLEN, José Antonio Pellicer. 2014. *Rediseño e implementación del módulo de Recursos Humanos de la herramienta GESPRO 13.05*. 2014.

ALVAREZ, Miguel Angel. 2014. ¿Qué es MVC? 2014. Url: ⟨http://www.desarrolloweb.com/articulos/que-es-mvc.html⟩.

CAKEPHP. 2016. *Entendiendo el Modelo - Vista - Controlador*. 2016. Dirección: ⟨http://book.cakephp.org/2.0/es/cakephp-overview/understanding-model-view-controller.html#beneficios⟩.

CERDA, Aníbal Sayid Valdivia. 2003. *Modelo de Diseño en UML e Implantación del Sistema de Gestión de Detallado Telefónico para las dependencias de la Universidad de Colima*. 2003.

CIENCIAS AGRARIAS, Facultad de. 2015. Modelo Entidad-Relación. 2015, págs. 10. Url: ⟨http://www.fca.unl.edu.ar/agromatica/Docs/09-ModeloEntRel.PDF⟩.

CONTABLE, El rincón del. 2016. *Valor Residual*. 2016. Dirección: ⟨http://rincondelcontable.blogspot.com/2012/03/valor-residual.html⟩.

DEBITOOR. 2016. *Definición de Valor Residual*. 2016. Dirección: ⟨https://debitoor.es/glosario/definicion-valor-residual⟩.

DEPRECIACIÓN. 2016. *Depreciación Acumulada*. 2016. Dirección: ⟨http://depreciacion.net/acumulada/⟩.

FREEMAN, Peter. 1987. *A Perspective on Reusability*. 1987.

FUOC. 2014. Introducción al Lenguaje de Modelado Unificado (UML). 2014.

GALLEGO, Manuel Trigas. 2015. Gestión de proyectos informáticos. Metodología SCRUM. 2015, págs. 56.

GERENCIE. 2016. *Definición de Depreciación*. 2016. Dirección: ⟨http://www.gerencie.com/depreciacion.html⟩.

GESPRO_UCI. 2013. *Ecosistema GESPRO*. 2013. Dirección: ⟨http://suitegespro.blogspot.com/2013/02/normal-0-false-false-false-en-us-x-none.html⟩.

GUILLÉN, Diego F. 2007. *Ruby Fácil*. 2007.

GUILLERMO STORTI Gladys Ríos, Gabriel Campodónico. 2007. Base de datos: Modelo Entidad Relación. 2007, págs. 14.

GUTIÉRREZ, Javier J. 2007. ¿Qué es un framework web? 2007, págs. 4.

HENRIK PESTANO PINO, Pedro Yobanis Piñero Pérez. 2011. Líneas de productos de software, de la idea a la práctica. 2011, vol. 4, n.º 1.

ISO. 2014. *Norma Internacional ISO 55000, Gestión de Activos - Aspectos generales, principios y terminología*. Primera Edición. 2014.

JORRÍN, Michaell González. 2012. Conceptos generales asociados al repositorio de activos de software. 2012.

LLEDÓ, Pablo. 2015. ¿Qué estructura organizacional se recomienda para proyectos? *Entorno Económico de Cuyo*. 2015.

LÓPEZ, Francisco J Toro. 2012. *Gestión de proyectos con enfoque PMI: Project y Excel*. 2012.

MARC GIBERT GINESTÁ, Oscar Pérez Mora. 2002. Bases de datos en PostgreSQL. 2002.

MATSUMOTO, Yukihiro. 2001. *Ruby in a Nutshell*. 2001. 0-59600-214-9.

ORALLO, Enrique Hernández. 2014. El Lenguaje Unificado de Modelado (UML). 2014.

PALACIO, Juan. 2006. El modelo Scrum. 2006. Url: ⟨http://www.navegapolis.net⟩.

PALACIO, Juan. 2014. *Gestión de proyectos Scrum Manager(Scrum Manager I y II)*. Scrum Manager, 2014.

PARADIGM, Visual. 2016. *Visual Paradigm*. 2016. Dirección: ⟨https://www.visual-paradigm.com/support/documents/vpuserguide/12/13/5963_visualparadi.html⟩.

PMI. 2013. *Guía de los Fundamentos para la Dirección de Proyectos (Guía del PMBOK)*. Quinta edición. 2013.

POSTGRESQL. 2016. *PostgreSQL*. 2016. Dirección: ⟨https://www.postgresql.org/about/⟩.

PRESSMAN, Roger. 2005. *Ingeniería del Software: Un Enfoque Práctico*. Sexta, 2005.

PRESSMAN, Roger S. 2002. *Ingeniería de Software. Un enfoque práctico*. Quinta Edición, 2002.

PUPO, Iliana Pérez. 2011. *Propuesta de metodología para el diseño e implantación de repositorios de activos de software reutilizables.* 2011.

QUIÑONES, Karina Mileisis Torres. 2014. *Estrategia de formación integrada en Gestión de Proyectos Informáticos*. 2014.

RAE. 2016a. *Concepto de Depreciación*. 2016. Concepto de Depreciación. Dirección: ⟨http://dle.rae.es/?id=CFz1U2T⟩.

RAE. 2016b. *Concepto de Reutilizar*. 2016. Concepto de Reutilizar. Dirección: ⟨http://dle.rae.es/?id=WMGvvdn⟩.

SAM RUBY Dave Thomas, David Heinemeier Hansson. 2009. *Agile Web Development with Rails*. Third Edition. 2009.

SOMMERVILLE, Ian y GALIPIENSO, María Isabel Alfonso. 2005. *Ingeniería del software*. 2005.

TARGETWARE. 2013. *Visual Paradigm para UML*. 2013. Dirección: ⟨http://www.software.com.ar/p/visual-paradigm-para-uml⟩.

UBUNTU, Guía. 2008. *PgAdmin III*. 2008. Dirección: ⟨http://www.guia-ubuntu.com/index.php/PgAdmin_III⟩.

Generado con LaTeX: 30 de junio de 2016: 4:47pm

Apéndices

A.1. Procesos de la Gestión de Adquisiciones:

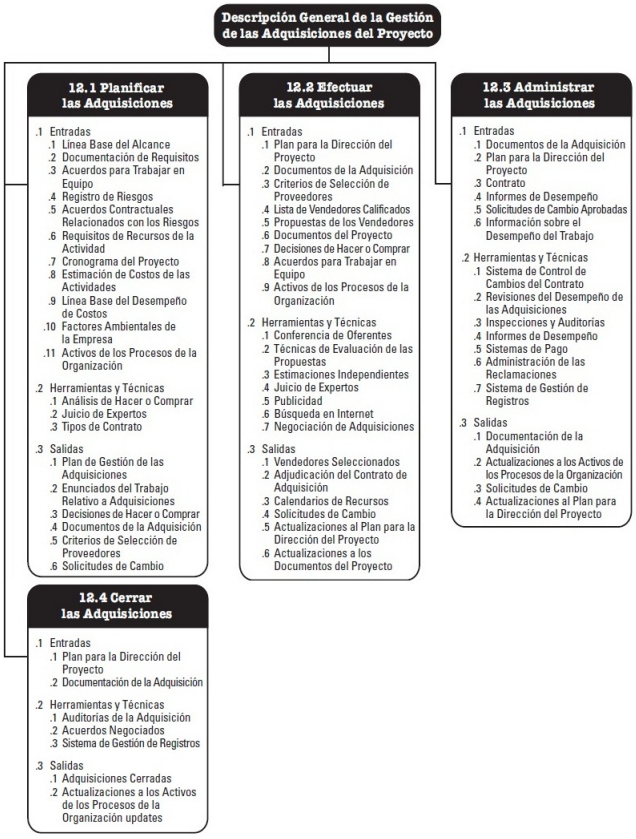
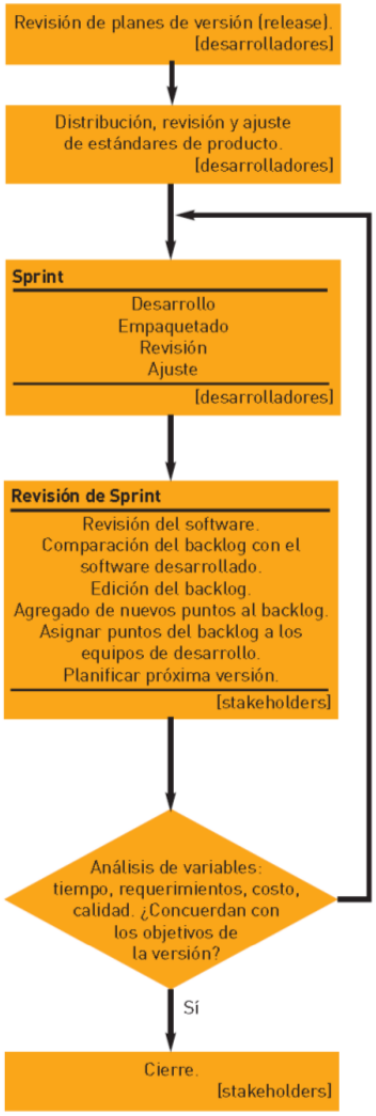


Figura A.1. Procesos de la Gestión de Adquisiciones según PMBOK (PMI, 2013)

A.2. Fases de la metodología SCRUM



REVISIÓN DE PLANES DE VERSIÓN:
Se revisa que hay que hacer y en que punto está la distribución actual.

SPRINT:
Es la fase de desarrollo iterativa.
Desarrollo: Análisis, implementación, testing.
Empaquetar: Generar paquetes ejecutables
Revisión: Resolución de problemas y se añaden nuevos ítems.
Ajustes: Uso de los ajustes para mejorar el producto.

SPRINT REVIEW:
Después del Sprint se hace una reunión con el ScrumMaster donde se revisa el producto del Sprint anterior y en el que se pueden añadir puntos nuevos al backlog.

Cierre:
En esta fase se encuentran las típicas actividades de fin de proyecto como, hacer una versión distribuible, testear, marketing etc....

Figura A.2. Fases de SCRUM (GALLEGO, 2015)

APÉNDICE B

Propuesta de solución

B.1. Empleo de la gema whenever en el módulo propuesto

Instalación en GESPRO de la gema whenever:

Abrir consola y colocar los siguientes comandos:

```
$ gem install dir_gema/whenever
$ cd gespro
$ wheneverize .
Este comando crea un archivo inicial en config/schedule.rb de gespro.

Luego se abre el archivo schedule.rb generado y se introduce el siguiente fragmento
de código:

every '00 00 01 * *' do
   rake "gespro:gespro_execute_depreciation"
end

Este fragmento define que se va a ejecutar la tarea gespro_execute_depreciation todos
los días primeros de cada mes.

Posterioremente se "traduce" el archivo schedule.rb al crontab de la siguiente forma:
# bundle exec whenever –update-crontab

Si se quiere ver el resultado que proporciona el crontab sin actualizar:

# bundle exec whenever
```

B.1.1. Tarea a ejecutar definida en el módulo propuesto

Código fuente B.1. Tarea gespro_execute_depreciation a ejecutar definida en el módulo propuesto para calcular la depreciación acumulada y el valor actual de los activos registrados en el sistema.

```
# -*- encoding: utf-8 -*-

require 'uri'
task "gespro:init" => :environment

 def depreciation_asset
   all_assets =  GesproAsset.all
   time_now =  Time.now
   all_assets.each do |asset|
    accumulated_depreciation = asset.accumulated_depreciation
    accumulated_depreciation += asset.cost * (asset.depreciation_rate
       /100)
      actual_value = asset.cost - accumulated_depreciation
      asset.accumulated_depreciation = accumulated_depreciation
      asset.actual_value = actual_value
      asset.last_depreciation = time_now
     asset.save
    end
 end


namespace :gespro do
desc <<-DESC
    Execute depreciation assets
  DESC
  task :gespro_execute_depreciation => :init do
    depreciation_asset
  end
end
```

B.2. Historias de usuarios asociadas a los requisitos funcionales

Tabla B.1. Historia de usuario # 3

Historia de usuario	
Número: 3	**Nombre:** Eliminar activo intangible a nivel de centro

Continúa en la próxima página

Tabla B.1. Continuación de la página anterior

Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Puntos estimados: 0.1	**Iteración asignada:** 1
Programador responsable: Ernesto Soler Calaña	
Descripción: Permite eliminar un activo intangible registrado a nivel de centro en el sistema. Si el usuario elimina correctamente el activo, el sistema debe mostrar un mensaje indicando que la operación fue realizada satisfactoriamente.	
Observaciones: En caso que el activo no sea eliminado, el sistema debe mostrar un mensaje de error con la explicación del mismo.	

Tabla B.2. Historia de usuario # 4

Historia de usuario	
Número: 4	**Nombre:** Crear activo físico a nivel de centro
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 0.8	**Iteración asignada:** 1
Programador responsable: Ernesto Soler Calaña	
Descripción: Permite crear un nuevo activo físico a nivel de centro e incluirlo en el sistema, luego de haber llenado los datos correspondientes. En caso de que se cree correctamente el activo, el sistema debe mostrar un mensaje indicando que la operación fue realizada satisfactoriamente.	
Observaciones: En caso que el activo no sea creado correctamente, el sistema debe mostrar un mensaje de error con la explicación del mismo. Si el usuario no introduce los campos necesarios para la correcta creación del activo intangible, el sistema debe mostrar un mensaje de error, indicando los campos que faltan por llenar.	

Tabla B.3. Historia de usuario # 5

Historia de usuario	
Número: 5	**Nombre:** Modificar activo físico a nivel de centro
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 0.5	**Iteración asignada:** 1
Programador responsable: Ernesto Soler Calaña	
Descripción: Permite modificar un activo físico registrado a nivel de centro en el sistema. Si el usuario modifica correctamente el activo, el sistema debe mostrar un mensaje indicando que la operación fue realizada satisfactoriamente.	
Observaciones: En caso que el activo no sea modificado, el sistema debe mostrar un mensaje de error con la explicación del mismo. Si el usuario no introduce los campos necesarios para la correcta modificación del activo intangible, el sistema debe mostrar un mensaje de error, indicando los campos que faltan por llenar.	

Tabla B.4. Historia de usuario # 6

Historia de usuario	
Número: 6	**Nombre:** Eliminar activo físico a nivel de centro
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Puntos estimados: 0.1	**Iteración asignada:** 1
Programador responsable: Ernesto Soler Calaña	
Descripción: Permite eliminar un activo físico registrado a nivel de centro en el sistema. Si el usuario elimina correctamente el activo, el sistema debe mostrar un mensaje indicando que la operación fue realizada satisfactoriamente.	
Observaciones: En caso que el activo físico no sea eliminado, el sistema debe mostrar un mensaje de error con la explicación del mismo.	

Tabla B.5. Historia de usuario # 7

Historia de usuario	
Número: 7	**Nombre:** Mostrar activo intangible a nivel de centro
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Puntos estimados: 0.2	**Iteración asignada:** 1
Programador responsable: Ernesto Soler Calaña	
Descripción: Permite mostrar el contenido asociado a un activo intangible registrado a nivel de centro en el sistema.	
Observaciones: En caso que la información asociada al activo no sea mostrada, el sistema debe mostrar un mensaje de error con la explicación del mismo.	

Tabla B.6. Historia de usuario # 8

Historia de usuario	
Número: 8	**Nombre:** Mostrar activo físico a nivel de centro
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Puntos estimados: 0.2	**Iteración asignada:** 1
Programador responsable: Ernesto Soler Calaña	
Descripción: Permite mostrar el contenido asociado a un activo físico registrado a nivel de centro en el sistema.	
Observaciones: En caso que la información asociada al activo no sea mostrada, el sistema debe mostrar un mensaje de error con la explicación del mismo.	

Tabla B.7. Historia de usuario # 9

Historia de usuario	
Número: 9	**Nombre:** Filtrar por categorías de activos registrados a nivel de centro
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 0.2	**Iteración asignada:** 1
Programador responsable: Ernesto Soler Calaña	
Descripción: Permite filtrar los activos registrados a nivel de centro por sus categorías. Si se realiza la operación correctamente el sistema debe mostrar una tabla con los activos que cumplan con la opción de filtrado.	
Observaciones: En caso que no existan coincidencias a la hora de filtrar según la categoría seleccionada, el sistema debe mostrar un mensaje indicando que no existen activos con dicha categoría.	

Tabla B.8. Historia de usuario # 10

Historia de usuario	
Número: 10	**Nombre:** Filtrar por estados de activos registrados a nivel de centro
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 0.2	**Iteración asignada:** 1
Programador responsable: Ernesto Soler Calaña	
Descripción: Permite filtar los activos registrados a nivel de centro por sus estados. Si se realiza la operación correctamente el sistema debe mostrar una tabla con los activos que cumplan con la opción de filtrado.	
Observaciones: En caso que no existan coincidencias a la hora de filtrar según el estado seleccionado, el sistema debe mostrar un mensaje indicando que no existen activos con dicho estado.	

Tabla B.9. Historia de usuario # 11

Historia de usuario	
Número: 11	**Nombre:** Filtrar Activos por Proyecto
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 0.2	**Iteración asignada:** 1
Programador responsable: Ernesto Soler Calaña	
Descripción: Permite filtrar los activos por su proyecto asociado. Si se realiza la operación correctamente el sistema debe mostrar una tabla con los activos que cumplan con la opción de filtrado.	
Observaciones: En caso que no existan coincidencias a la hora de filtrar según el proyecto seleccionado, el sistema debe mostrar un mensaje indicando que no existen activos con dicho estado. Este filtrado solo se ejecuta a nivel de Centro.	

Tabla B.10. Historia de usuario # 12

Historia de usuario	
Número: 12	**Nombre:** Crear activo intangible a nivel de proyecto
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 0.8	**Iteración asignada:** 2
Programador responsable: Ernesto Soler Calaña	
Descripción: Permite crear un nuevo activo intangible a nivel de proyecto e incluirlo en el sistema, luego de haber llenado los datos correspondientes. En caso que se cree correctamente el activo, el sistema debe mostrar un mensaje indicando que la operación fue realizada satisfactoriamente.	
Observaciones: En caso que el activo no sea creado correctamente, el sistema debe mostrar un mensaje de error con la explicación del mismo. Si el usuario no introduce los campos necesarios para la correcta creación del activo intangible, el sistema debe mostrar un mensaje de error, indicando los campos que faltan por llenar.	

Tabla B.11. Historia de usuario # 13

Historia de usuario	
Número: 13	**Nombre:** Modificar activo intangible intangible a nivel de proyecto
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 0.5	**Iteración asignada:** 2
Programador responsable: Ernesto Soler Calaña	
Descripción: Permite modificar un activo intangible registrado a nivel de proyecto en el sistema. Si el usuario modifica correctamente el activo, el sistema debe mostrar un mensaje indicando que la operación fue realizada satisfactoriamente.	
Observaciones: En caso que el activo no sea modificado, el sistema debe mostrar un mensaje de error con la explicación del mismo. Si el usuario no introduce los campos necesarios para la correcta modificación del activo intangible, el sistema debe mostrar un mensaje de error, indicando los campos que faltan por llenar.	

Tabla B.12. Historia de usuario # 14

Historia de usuario	
Número: 14	**Nombre:** Eliminar activo intangible a nivel de proyecto
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Puntos estimados: 0.1	**Iteración asignada:** 2
Programador responsable: Ernesto Soler Calaña	
Descripción: Permite eliminar un activo intangible registrado a nivel de proyecto en el sistema. Si el usuario elimina correctamente el activo, el sistema debe mostrar un mensaje indicando que la operación fue realizada satisfactoriamente.	

Continúa en la próxima página

Tabla B.12. Continuación de la página anterior

Observaciones: En caso que el activo no sea eliminado, el sistema debe mostrar un mensaje de error con la explicación del mismo.

Tabla B.13. Historia de usuario # 15

Historia de usuario	
Número: 15	**Nombre:** Mostrar activo intangible a nivel de proyecto
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Puntos estimados: 0.2	**Iteración asignada:** 2
Programador responsable: Ernesto Soler Calaña	
Descripción: Permite mostrar el contenido asociado a un activo intangible registrado a nivel de proyecto en el sistema.	
Observaciones: En caso que la información asociada al activo no sea mostrada, el sistema debe mostrar un mensaje de error con la explicación del mismo.	

Tabla B.14. Historia de usuario # 16

Historia de usuario	
Número: 16	**Nombre:** Crear activo físico a nivel de proyecto
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 0.8	**Iteración asignada:** 2
Programador responsable: Ernesto Soler Calaña	
Descripción: Permite crear un nuevo activo físico a nivel de proyecto e incluirlo en el sistema, luego de haber llenado los datos correspondientes. En caso que se cree correctamente el activo, el sistema debe mostrar un mensaje indicando que la operación fue realizada satisfactoriamente.	
Observaciones: En caso que el activo no sea creado correctamente, el sistema debe mostrar un mensaje de error con la explicación del mismo. Si el usuario no introduce los campos necesarios para la correcta creación del activo intangible, el sistema debe mostrar un mensaje de error, indicando los campos que faltan por llenar.	

Tabla B.15. Historia de usuario # 17

Historia de usuario	
Número: 17	**Nombre:** Modificar activo físico a nivel de proyecto
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 0.5	**Iteración asignada:** 2
Programador responsable: Ernesto Soler Calaña	

Continúa en la próxima página

Tabla B.15. Continuación de la página anterior

Descripción: Permite modificar un activo físico registrado a nivel de proyecto en el sistema. Si el usario modifica correctamente el activo, el sistema debe mostrar un mensaje indicando que la operación fue realizada satisfactoriamente.
Observaciones: En caso que el activo no sea modificado, el sistema debe mostrar un mensaje de error con la explicación del mismo. Si el usuario no introduce los campos necesarios para la correcta modificación del activo intangible, el sistema debe mostrar un mensaje de error, indicando los campos que faltan por llenar.

Tabla B.16. Historia de usuario # 18

Historia de usuario	
Número: 18	**Nombre:** Eliminar activo físico a nivel de proyecto
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Puntos estimados: 0.1	**Iteración asignada:** 2
Programador responsable: Ernesto Soler Calaña	
Descripción: Permite eliminar un activo físico registrado a nivel de proyecto en el sistema. Si el usario elimina correctamente el activo, el sistema debe mostrar un mensaje indicando que la operación fue realizada satisfactoriamente.	
Observaciones: En caso que el activo físico no sea eliminado, el sistema debe mostrar un mensaje de error con la explicación del mismo.	

Tabla B.17. Historia de usuario # 19

Historia de usuario	
Número: 19	**Nombre:** Mostrar activo físico a nivel de proyecto
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Puntos estimados: 0.2	**Iteración asignada:** 2
Programador responsable: Ernesto Soler Calaña	
Descripción: Permite mostrar el contenido asociado a un activo físico registrado a nivel de proyecto en el sistema.	
Observaciones: En caso que la información asociada al activo no sea mostrada, el sistema debe mostrar un mensaje de error con la explicación del mismo.	

Tabla B.18. Historia de usuario # 20

Historia de usuario	
Número: 20	**Nombre:** Filtrar por categorías de activos registrados a nivel de proyecto
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja

Continúa en la próxima página

Tabla B.18. Continuación de la página anterior

Puntos estimados: 0.2	**Iteración asignada:** 2
Programador responsable: Ernesto Soler Calaña	
Descripción: Permite filtrar los activos registrados a nivel de proyecto por sus categorías. Si se realiza la operación correctamente el sistema debe mostrar una tabla con los activos que cumplan con la opción de filtrado.	
Observaciones: En caso que no existan coincidencias a la hora de filtar según la categoría seleccionada, el sistema debe mostrar un mensaje indicando que no existen activos con dicha categoría.	

Tabla B.19. Historia de usuario # 21

Historia de usuario	
Número: 21	**Nombre:** Filtrar por estados de activos registrados a nivel de proyecto
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 0.2	**Iteración asignada:** 2
Programador responsable: Ernesto Soler Calaña	
Descripción: Permite filtrar los activos registrados a nivel de proyecto por sus estados. Si se realiza la operación correctamente el sistema debe mostrar una tabla con los activos que cumplan con la opción de filtrado.	
Observaciones: En caso que no existan coincidencias a la hora de filtrar según el estado seleccionado, el sistema debe mostrar un mensaje indicando que no existen activos con dicho estado.	

Tabla B.20. Historia de usuario # 22

Historia de usuario	
Número: 22	**Nombre:** Calcular Valor Actual y Depreciación Acumulada para todos los activos.
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 0.9	**Iteración asignada:** 3
Programador responsable: Ernesto Soler Calaña	
Descripción: Permite calcular el Valor Actual y la Depreciación Acumulada de todos los activos registrados en el sistema.	
Observaciones: La operación se ejecuta automáticamente todos los días primero de cada mes a las 12:00 am.	

Tabla B.21. Historia de usuario # 23

Historia de usuario	
Número: 23	**Nombre:** Permitir la dependencia entre activos
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 0.9	**Iteración asignada:** 3

Continúa en la próxima página

Tabla B.21. Continuación de la página anterior

Programador responsable: Ernesto Soler Calaña
Descripción: Permite realizar dependencias entre activos registrados en el sistema sin importar el tipo (intangible o físico). Si el usuario realiza la dependecia correctamente un activo con uno o varios activos, el sistema debe mostrar un mensaje indicando que la operación fue realizada satisfactoriamente.
Observaciones: En caso que no se se logre realizar la operación, el sistema debe mostrar un mensaje de error con la descripción del mismo.

Tabla B.22. Historia de usuario # 24

Historia de usuario	
Número: 24	**Nombre:** Mostrar los activos dependientes
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 0.9	**Iteración asignada:** 3
Programador responsable: Ernesto Soler Calaña	
Descripción: Muestra los activos dependientes de un activo seleccionado.	
Observaciones: En caso que no se se logre realizar la operación, el sistema debe mostrar un mensaje de error con la descripción del mismo.	

Tabla B.23. Historia de usuario # 25

Historia de usuario	
Número: 25	**Nombre:** Eliminar activos dependientes
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 0.9	**Iteración asignada:** 3
Programador responsable: Ernesto Soler Calaña	
Descripción: Permite eliminar los activos dependientes de un activo en particular.	
Observaciones: En caso que no se se logre realizar la operación, el sistema debe mostrar un mensaje de error con la descripción del mismo.	

Tabla B.24. Historia de usuario # 26

Historia de usuario	
Número: 26	**Nombre:** Buscar activo por código
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Puntos estimados: 0.5	**Iteración asignada:** 3
Programador responsable: Ernesto Soler Calaña	

Continúa en la próxima página

Tabla B.24. Continuación de la página anterior

Descripción: Permite buscar un activo por el código entrado. Si se realiza la operación correctamente el sistema debe mostrar una tabla con los activos que cumplan o se asemejen con la opción de búsqueda.
Observaciones: En caso que no existan coincidencias a la hora de buscar según el código introducido, el sistema debe mostrar un mensaje indicando que no existen activos con dicho código.

Tabla B.25. Historia de usuario # 27

Historia de usuario	
Número: 27	**Nombre:** Buscar activo por título
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Puntos estimados: 0.5	**Iteración asignada:** 3
Programador responsable: Ernesto Soler Calaña	
Descripción: Permite buscar un activo por el título entrado. Si se realiza la operación correctamente el sistema debe mostrar una tabla con los activos que cumplan o se asemejen con la opción de búsqueda.	
Observaciones: En caso que no existan coincidencias a la hora de buscar según el título introducido, el sistema debe mostrar un mensaje indicando que no existen activos con dicho título.	

Tabla B.26. Historia de usuario # 28

Historia de usuario	
Número: 28	**Nombre:** Buscar activo por descripción
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Puntos estimados: 0.5	**Iteración asignada:** 3
Programador responsable: Ernesto Soler Calaña	
Descripción: Permite buscar un activo por su descripción. Si se realiza la operación correctamente el sistema debe mostrar una tabla con los activos que cumplan o se asemejen con la opción de búsqueda.	
Observaciones: En caso que no existan coincidencias a la hora de buscar según el título introducido, el sistema debe mostrar un mensaje indicando que no existen activos con dicha descripción.	

APÉNDICE C

Resultados

C.1. Pruebas de caja negra

Ver la carpeta pruebas de caja negra adjunta a esta investigación, en la cual se encuentran las pruebas sometidas a las interfaces del módulo.