Universidad de las Ciencias Informáticas
Facultad 3

Plugin para importar tareas en el Redmine desde otras herramientas de planificación de proyectos.

Trabajo de Diploma para optar por el título de

Ingeniero en ciencias Informáticas

Autor: Luis Enrique Torrado Ruiz.

Tutores: Msc. César Lage Codorníu.

Msc. Nemury Silega Martínez.

La Habana, Cuba
Junio del 2011

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____

Luis Enrique Torrado Ruíz

_____ _____

Msc. César Lage Codorníu Msc. Nemury Silega Martínez

Datos de contacto

Msc. César Lage Codorníu

Correo electrónico: clage@uci.cu

Msc. Nemury Silega Martínez

Correo electrónico: nsilega@uci.cu

Agradecimientos

A mi familia por apoyarme en todo momento e impulsarme a seguir
A mis tutores por toda la ayuda que me brindaron en el desarrollo del trabajo
A mi novia por ser tan especial
A mis amigos de la vocacional Fidel, Gustavo, Leitniz, Karel, Ariel y Jorge.
A mis amigos de la universidad aquellos que me permitieron compartir una parte de mi vida con ellos en especial a Álvaro, Rolando, Sandy, Jorge Carlos y Aliankis.

Dedicatoria

A mis bisabuelas Delia y Milagros por ser los pilares de mi familia y ejemplo amor y bondad
A mis abuelos Ada y Evelio por ser mis ángeles de la guarda y mis paradigmas de personas.
A mis padres por el apoyo que siempre me han brindado
A mis hermanos por el cariño que me brindan y lo que significan en mi vida
A mi amigo Sandy Azorín por compartir mis sueños

Resumen

La tarea de planificar y gestionar proyectos se ha ido desarrollando con el creciente flujo de información que circula por los mismos, por lo que se han creado aplicaciones para la realización de este trabajo. Este es el caso del Redmine, el que realiza esta función a través de una interfaz web. Sin embargo esta aplicación no brinda opciones de interoperabilidad entre esta y otras herramientas de planificación. Actualmente la tendencia para la resolución de los problemas del Redmine ha sido la creación de *plugins*[1] a terceros o para el mismo sistema.

El presente trabajo tiene como objetivo el diseño e implementación de un *plugin* que permita importar elementos generados por otras aplicaciones de planificación de proyectos al Redmine. Para llevar a cabo el trabajo se realiza un diseño que cumple con los parámetros establecidos por la arquitectura y se emplean las herramientas y tecnologías seleccionadas para la implementación. Además se realiza un periodo de pruebas para la validación del trabajo asegurándose del que el sistema cumpla con las normas vigentes para el desarrollo.

Se presentan como resultados un sistema preparado para su instalación y funcionamiento.

Palabras clave:

Redmine, plugin, interfaz, web, interoperabilidad

[1] Plugin: Este concepto se encuentra definido en el capítulo 2 epígrafe 2.2.1

Tabla de contenido

Tabla de Contenidos

Tabla de Figuras

Introducción

Los proyectos informáticos han ido evolucionando a la par de las tecnologías de la informática y las comunicaciones, ha aumentado la envergadura de los mismos y la complejidad, haciéndose cada vez mayor el flujo de información a procesar, siendo estos más difícil de planificar y gestionar.

Existen aplicaciones que se encargan de gestionar proyectos como: Redmine, el cual es un gestor y planificador con interfaz web, orientado a la coordinación de tareas, comunicación de participantes, y que puede especializarse en proyectos de desarrollo gracias a elementos como: La integración en un repositorio de código. (1)

Entre las funcionalidades del Redmine se encuentran: la gestión múltiples proyectos, personalización de los proyectos, activación y desactivación de módulos, sistema flexible del seguimiento de tareas, notificaciones y exportación en distintos formatos.

Para la gestión de proyecto al Redmine se le han agregado funcionalidades afines a las necesidades de los diferentes centros de desarrollo de la Universidad de las Ciencias Informáticas (UCI). A través de *plugin*s se han suplido algunas de las necesidades inmediatas de los proyectos teniendo en cuenta el cumplimiento de las políticas de la UCI.

El uso del Redmine para la gestión y planificación de proyectos es una apolítica de nuestra universidad, sin embargo, muchos planificadores prefieren realizar su trabajo en otras herramientas como son Microsoft Excel y Microsoft Project. Aun cuando el redmine presenta variantes como son el calendario y el diagrama de Gantt. Esto se debe a que estas aplicaciones de Microsoft brindan opciones para el manejo y seguimiento de las actividades que no presenta el redmine, además son herramientas de escritorio. Los elementos generados por estas herramientas se pueden importar al redmine pero no existe una interoperabilidad entre estos y agregar las tareas que se encuentran dentro de los elemento importados de forma manual llevaría mucho tiempo y esfuerzo. A partir de esta situación problemática se plantea como **problema a resolver:** El Redmine no brinda todas las funcionalidades requeridas para el desglose de trabajo y el establecimiento de la secuencia de actividades dentro de la planificación de proyectos.

Se define como **objeto de estudio** interoperabilidad entre herramientas de gestión de proyectos.

Se plantea como **objetivo general** desarrollar un *plugin* para importar tareas en el Redmine desde otras herramientas de planificación de proyectos. El trabajo se

encuentra enmarcado dentro **campo de acción** Diseño e implementación de aplicaciones para la interoperabilidad entre herramientas de gestión de proyecto
La investigación tiene como **objetivos específicos:**

- Identificar las posibles herramientas de planificación de proyectos para la comunicación y las posibilidades de interoperabilidad que brinda el Redmine.
- Definir el marco teórico de la investigación
- Diseñar e implementar un *plugin* para importar tareas en el Redmine.
- Evaluar los resultados obtenidos.

Con el fin de cumplimiento a los objetivos específicos se definieron como **tareas de investigación**
Estudio sobre las diferentes herramientas de planificación de proyectos.

- Realización de la fundamentación teórica conceptual de la investigación.
- Análisis sobre la arquitectura del Redmine y opciones de interoperabilidad.
- Caracterización de principales tecnologías y patrones a utilizar para el desarrollo de software
- Diseño de las funcionalidades del plugin
- Diseño de los diagramas de clases a partir del a análisis y el levantamiento de requisitos.
- implementación del plugin a partir de del diseño elaborado
- Elaboración de pruebas para la evaluación de los componentes
- Resolución de no conformidades detectadas
- Análisis del comportamiento de la herramienta una vez en funcionamiento.
-

Para el diseño de la investigación se plantea la siguiente **idea a defender:** con el diseño e implementación de un *plugin* se logrará importar tareas de otras herramientas de gestión y planificación de proyectos hacia el Redmine complementando la interoperabilidad de las mismas.
Para dar cumplimiento a los objetivos propuestos se utilizaron los siguientes **métodos de investigación.**

Métodos Teóricos

- **Histórico – Lógico:** Para realizar un estudio de las tendencias históricas y actuales en el desarrollo de herramientas de gestión y planificación de proyectos.
- **Analítico – Sintético:** Para realizar un análisis de la información empleada

para la investigación, así como la interoperabilidad de las herramientas de gestión y planificación de proyectos.

- **Modelación:** Para modelar los requerimientos funcionales que se proponen en la solución y el diseño de los nuevos componentes visuales.

Métodos Empíricos

- **Entrevistas:** Para realizar el estudio de los procesos actuales que se desarrollan en la universidad sobre las herramientas de gestión y planificación de proyectos.
- **Experimento:** Para establecer las diferentes pruebas de calidad con el propósito de determinar la fiabilidad del sistema y el mejoramiento de la usabilidad del mismo, a través de la detección de errores.

Como **posibles resultados** se espera obtener un *plugin* para importar tareas en el Redmine desde otras herramientas de planificación de proyectos y la documentación vinculada a la solución, además que este brinde todas las funcionalidades requeridas para el desglose de trabajo y el establecimiento de la secuencia de actividades dentro de la planificación de proyectos.

Estructura del documento

El trabajo de diploma cuenta con Resumen, Introducción, Tres Capítulos, Conclusiones, Recomendaciones, Referencias bibliográficas, Bibliografía y Glosario de Términos.

Capítulo 1 Fundamentación teórica, se realiza un estudio sobre la interoperabilidad, conceptos y formas de interoperabilidad, y opciones de interoperabilidad para el Redmine. Se muestran cuales son las principales características y deficiencias del Redmine así como las herramientas con las que se desea interoperar. También se abordan las principales herramientas para el desarrollo.

Capítulo 2 Características y diseño del sistema, se presenta la propuesta de solución que se quiere implementar, se diseña el modelo de dominio se realiza el proceso de captura de requisitos. Diseño del sistema, se lleva a cabo el diseño del *plugin*, se describe y muestra el diagrama de clases del diseño, además, se realiza el diagrama de despliegue el cual describe cómo y dónde el *plugin* será puesto en funcionamiento

Capítulo 3 Implementación y prueba, se realiza todo lo relacionado con los flujos de trabajo de implementación y prueba, se desarrolla el diagrama de implementación para dar una visión de cómo las clases, artefactos y otros elementos así como las

conexiones entre ellos y se realizan los casos de pruebas para garantizar buena calidad al software.

Capítulo 1. Fundamentación teórica

1.1 Introducción

En la actualidad existen diversos sistemas operativos, cada cual con sus defensores y detractores y gran gama de software desarrolladas en múltiples entornos, tanto privativos como libres. Una de las principales necesidades de los usuarios es que estos sistemas y aplicaciones se interconecten para que la comunicación fluya y sea beneficioso para todos.

A través de este conflicto surge por primera vez la palabra interoperabilidad entre aplicaciones, tecnologías y sistemas. (2)

En este capítulo se presenta un estudio sobre la interoperabilidad de los sistemas informáticos, además de las opciones que brindan los sistemas de gestión de proyectos haciendo énfasis en el Redmine. También se abordan temas como la planificación de proyectos y los elementos que esta genera, que son los que se tiene como objetivo interoperar con el Redmine.

1.2 Interoperabilidad

La mayoría de las organizaciones disponen de sistemas informáticos de diferentes fabricantes y dedicados a distintas operaciones, diversidad que también se da en las bases de datos. Como resultado de esta situación, las empresas se ven obligadas a mantener más de un sistema para la realización de una tarea, siendo difícil la extracción de toda la información necesaria, puesto que esta se encuentra distribuida en aplicaciones distintas. Las tecnologías de integración se encargan de que diferentes sistemas informáticos sean capaces de comunicarse y compartir datos.

1.2.1 Conceptos

El Diccionario de la Real Academia de la Lengua Española no define ese término. Una de las primeras definiciones aparece como Interoperabilidad militar. La cual la plantea como “La habilidad de los sistemas, unidades o fuerzas para proveer servicios a y para aceptar servicios de otros sistemas, unidades o fuerzas, y para usar los servicios así intercambiados para operar efectivamente juntos”. (2)

Una de las primeras definiciones de interoperabilidad limitada a los servicios de administración electrónica aparece en el Acta sobre administración electrónica de los Estados Unidos de 2002 “que la define como la capacidad de diferentes sistemas operativos y software, aplicaciones y servicios para comunicar e intercambiar datos de una forma precisa, efectiva y consistente. Se trata de una definición formulada desde un punto de vista exclusivamente técnico”. (3)

También se puede encontrar la interoperabilidad definida como “La capacidad de los sistemas de tecnologías de la información y de las comunicaciones (TIC) y de los

procesos empresariales a los que apoyen, de intercambiar datos y posibilitar la puesta en común de información y conocimientos.” (4)

Por lo que se puede concluir que la interoperabilidad consiste en que sistemas heterogéneos dispongan de mecanismos que permitan intercambiar procesos y/o datos. En el entorno web, la interoperabilidad es una de las condiciones necesarias para que los usuarios tengan un completo acceso a la información disponible. Con el desarrollo actual de la informática la interoperabilidad entre aplicaciones es un requisito fundamental para el funcionamiento de los sistemas.

1.2.2 Formas de interoperabilidad

Existen diferentes formas de clasificar la interoperabilidad.

1. Para el caso específico de la web, dado que es una red altamente distribuida donde cada vez más se encuentran, por una parte, sistemas de información heterogénea y, por otra, usuarios que demandan un completo acceso a la información disponible; se clasifica sobre la base de distintos niveles:

- Infraestructura
- Sintaxis
- Estructura
- Semántica.

Para el caso de las arquitecturas basadas en servicios, se clasifica en:

- Aplicación y herramienta
- Contenido
- Infraestructura
- Usuario.

3. Para los sistemas que apoyan la educación, se clasifica principalmente en dos:

- Infraestructura
- Contenido (5)

El concepto de interoperabilidad de la información, en el contexto de las políticas de información y de administración electrónica de la Unión Europea y España, aparece ligado la interoperabilidad semántica, que trata, sobre todo, aquellos aspectos relacionados con el tratamiento de la información electrónica. La interoperabilidad consiste en un proceso continuo, en el que cualquier obstáculo afecta negativamente a la posibilidad de despliegue de la aplicación y de la prestación del servicio correspondiente. Además integra la interoperabilidad en tres formas de la misma los que son interoperabilidad técnica, semántica y organizativa como se muestra en la Figura 1. (4)

Figura 1: Formas de interoperabilidad. (4)

La interoperabilidad de la información define como aspectos más importantes el sintáctico y el semántico.

- **Interoperabilidad sintáctica**: se basa en la codificación de datos, generalmente, mediante la utilización de lenguajes de marcado estándar para el desarrollo de sistemas, los modelos de gestión de documentos, y el uso de formatos y formas de presentación de la información adecuadas para cada tipo de usuario; accesibilidad web, y otros
- **Interoperabilidad semántica:** está destinada a la descripción de los recursos de información para facilitar el intercambio de información y la recuperación óptima por parte de los usuarios. En este ámbito se suelen utilizar un conjunto de herramientas para la representación del conocimiento contenido en los recursos, como son: los vocabularios controlados, los metadatos, las ontologías, los *topic maps,* y otros. (6)

  Se escogen estas dos últimas formas de interoperabilidad ya que son las más adecuadas para llevar a cabo en el presente trabajo además le permiten a la solución cumplir con las necesidades planteadas.

1.2.3 Interoperabilidad para el Redmine

Redmine se distribuye libremente bajo licencia GNU General *Public License* v2 (GPL). Por lo que ya se ha interoperado con otras herramientas como *Alfresco* el cual es un sistema de administración de contenidos libre. También con *Enterprise Architect* que es un producto comercial de *Sparx Systems* que se presenta como una plataforma para el desarrollo y modelado colaborativo de artefactos basados en UML 2.1 y similares.

En la UCI el Redmine se encuentra interoperado con *Alfresco* y *Excriba* para la gestión de la documentación. Sin embargo este brinda opciones de interoperabilidad tanto para software libre como privativo. El Redmine suple muchas de las necesidades de interoperabilidad a través de *plugin*[2] que se le agregan al mismo o a otras aplicaciones para la comunicación con estos.

También son interesantes los *plugin*s de terceros, extensiones que utilizan otras aplica ciones para integrarse con Redmine de alguna manera.Por ejemplo para entorno de desarrollo Eclipse, el navegador Firefox o para el teléfono iphone. Los que se pueden encontrar en http://www.redmine.org/wiki/redmine/ThirdPartyTools.

1.3 Redmine

Gestionar los proyectos es una tarea crítica. La necesidad de organizar y administrar los recursos, tiempos, actividades y alcance hacen de esta un trabajo complejo que se agudiza ante el surgimiento de variables que pueden influir en la realización de un proyecto. (7)

El Redmine es un sistema multi-plataforma e independiente de la base de datos, programado con el framework Ruby on Rails, *open source* con licencia GPL, con un interfaz limpio y unas funcionalidades asombrosas para gestión de proyectos.

1.3.1 Características

El Redmine presenta una serie de características que ha hecho de este uno de los gestores de proyectos de tipo Web con más aceptación, a continuación se brindan algunas de dichas características. (1)

- **Gestión de múltiples proyectos**

Redmine permite gestionar múltiples proyectos desde una sola interfaz con una ventana de navegador. La navegación es muy sencilla y se puede saltar y cambiar de proyecto en cualquier momento. Los proyectos pueden definirse como privados, en los

[2] Plugin: Este concepto se encuentra definido en el capítulo 2 epígrafe 2.2.1

que el administrador debe dar acceso a cada miembro, o públicos, visibles para todo el mundo. También en cada proyecto puede definirse varios subproyectos.

- **Personalización de proyectos**

Cada proyecto es totalmente personalizable, encontrando proyectos muy distintos entre sí según sus objetivos. Lo más importante son los módulos que se pueden desactivar y activar para cada proyecto aunque existen módulos comunes para todos.

- **Sistema flexible de seguimiento de tareas**

Una de las mecánicas más útiles para el desarrollo de un proyecto en Redmine son las peticiones y su visualización. Estas peticiones se dividen en tres tipos y pueden asignarse a un miembro del proyecto. Se puede indicar una fecha para el inicio de la tarea y para el fin e incluso llevar un control del tiempo y el porcentaje real realizado. Se pueden obtener informes o incidencias sobre las peticiones. Además dentro de un proyecto se pueden establecer versiones y asignar tareas a determinadas versiones.

- **Integración en repositorios de código**

Redmine puede integrarse con un repositorio de código (subversión, Git, CVS, entre otros) que se encuentre instalado en la misma máquina, tan solo hay que indicarle el directorio local. La aplicación sirve así de interfaz web para el seguimiento del desarrollo del proyecto. Pueden descargarse los ficheros, ver el historial, los cambios e incluso descargar un archivo a modo de parche para aplicar a código desactualizado. Presenta sistema de seguimiento de versiones, aunque no se puede actualizar los ficheros directamente.

- **Uso de calendario y diagrama de Gantt**

Redmine incluye un calendario para visualizar todas las peticiones a lo largo de un tiempo elegido, marcado claramente día de inicio y día de fin de cada petición. Igualmente ocurre con la vista en el diagrama de Gantt, que va marcando el porcentaje completado conforme avanzan los días. Las peticiones que se visualizan en ambos casos están sujetas a los filtros definidos por el usuario.

- **Notificaciones**

Configurando previamente el servidor de correo SMTP, Redmine permite enviar notificaciones por correo a todos los proyectos, definiendo antes los eventos que activan los avisos. Además cada usuario en su configuración puede elegir recibir notificaciones de cualquier evento, o solo las relacionadas con el. Puede configurarse

el servidor de correo entrante, permitiendo así actualizar peticiones simplemente por email e incluso crear nuevas peticiones

- **Exportación a distintos formatos**

Los informes de peticiones que pueden generarse añadiendo filtros, y que permiten visualizar las diferentes tareas de un proyecto, pueden exportarse en PDF o formato CSV, con la opción de imprimirlos posteriormente en un formato organizado. Las paginas Wiki en cambio pueden exportarse como HTML o TXT.

La aplicación puede integrarse perfectamente con cualquier tecnología de asistencia del sistema operativo y con cualquier opción relacionada con el navegador de internet. Pero no se cuenta con opciones para la comunicación con herramientas de planificación de proyectos (1)

1.3.2 Deficiencias

Redmine es un gestor de proyectos muy potente y maduro, a pesar de su "poco" tiempo de desarrollo, además está inspirado en trac, por lo que se está estamos ante una herramienta muy flexible de por sí, y bastante estable.

La cantidad de opciones, la detallada configuración, la inclusión de herramientas como el foro y una wiki, y la gran cantidad de extensiones hacen al Redmine muy completo, pero hay algunos aspectos donde la personalización podría ser mayor.

Otra deficiencia es que las notificaciones por correo están adheridas a todos los proyectos creados, pueden activarse o desactivarse, pero tampoco puede personalizar individualmente a cada proyecto.

También se ha encontrado un fallo y es que las RSS provenientes de la wiki no funcionan correctamente no se puede leer su contenido en lectores de RSS. Además en el módulo de "Actividad", no se guardan los cambios realizados sobre el contenido que se quiere visualizar exactamente, volviendo siempre a mostrar la actividad por defecto cuando se vuelve al módulo.

Una de las principales deficiencias de Redmine queda en la instalación, que puede resultar muy compleja, no solo por los paquetes requeridos, sino por los pasos que se debe seguir para la realización de esta. Existen paquetes .deb que facilitan el proceso, pero no están bien documentados quedando un paso por realizar a mano para ejecutar la aplicación. Aunque hay mucha documentación al respecto, es demasiado informal y no siempre da resultados en todos los casos. (1)

Además el Redmine no brinda todas las funcionalidades requeridas para el desglose de trabajo y el establecimiento de la secuencia de actividades dentro de la planificación de proyectos. Y no interopera con otras herramientas de gestión y planificación de proyectos que realizan esta función.

1.4 Herramientas a interoperar

Se desea interoperar con el Redmine algunas de las herramientas usadas para la planificación de de los proyectos como son Microsoft Excel y Microsoft Project. Estas son usadas por las facilidades que brinda tanto de manejo como de desarrollo, y permite que se administre y coordine el trabajo con eficacia

1.4.1 Microsoft Excel

Microsoft Excel es una hoja de cálculo comercial por escrito y distribuido por Microsoft para Microsoft Windows y Mac OS X. Cuenta con cálculo, herramientas gráficas, tablas dinámicas y un lenguaje de programación de macros llamado Visual Basic para Aplicaciones. Ha sido una hoja de cálculo muy ampliamente aplicado para estas plataformas, especialmente desde la versión 5 en 1993. Excel forma parte de Microsoft Office. Las versiones actuales son compatibles con Mac, Windows 2010 y Windows 2011. (8)

Microsoft Office Excel es una herramienta eficaz que puede usar para crear y aplicar formato a hojas de cálculo, y para analizar y compartir información para tomar decisiones mejor fundadas. La interfaz de usuario de Microsoft Office Fluent, la visualización de datos enriquecida y las vistas de tabla dinámica permiten crear, de un modo más sencillo, gráficos de aspecto profesional y fácil uso. Office Excel, ofrece mejoras significativas para compartir datos con más seguridad. Pueden compartir información confidencial de la empresa de un modo más amplio y seguro con sus compañeros de trabajo, clientes y socios empresariales. (8)

1.4.2 Microsoft Project

Microsoft Project (o MSP o WINPROJ) software de gestión de proyectos desarrollado y vendido por Microsoft que está diseñado para ayudar a los gestores del proyecto en el desarrollo de planes. Este permite organizar la información acerca de la asignación de tiempos a las tareas, los costes asociados y los recursos, tanto de trabajo como materiales, del proyecto para que se puedan respetar los plazos sin exceder el presupuesto y conseguir así los objetivos planteados. (9)

Microsoft Project es la tercera aplicación basada en Windows, que en un par de años de su lanzamiento se convirtió en la herramienta gestión de proyectos de software dominante basado en PC. (9)

Aunque la marca como un miembro de la familia de Microsoft Office, nunca ha sido incluido en ninguna de las suites de oficina. Este fue también el caso de Office 2010. Se encuentra disponible actualmente en dos ediciones, Estándar y Profesional. Formato de archivo propietario de MS Project es. mpp. Microsoft Project y Microsoft

Project Server son los pilares de Microsoft Office Enterprise Project Management (EPM) de productos. (9)

1.5 Tecnologías y herramientas para desarrollo

A lo largo de la vida de un proyecto de desarrollo de Software, el programador puede utilizar diversas tecnologías y herramientas, las cuales muchas veces son definidas por supervisores o compartidas por expertos. A continuación se explican las utilizadas para la realización del plugin.

1.5.1 Lenguaje

Ruby es un lenguaje de scripts, multiplataforma, netamente orientado a objetos es software libre, fue creado por Yukihiro Matsumoto conocido como *Matz*. La primera versión fue liberada en 1995, hereda varias características de lenguajes como: *Perl*, *Smalltalk*, *Eiffel*, *Ada* y *Lisp*. Como lo indica su propio autor, es un lenguaje "aparentemente sencillo pero internamente complejo".

Esto quiere decir que mientras más el programador se abstrae en el paradigma orientado a objetos se notará realmente la complejidad del lenguaje, el cual se considera muy intuitivo casi a un nivel humano.

Ruby fue diseñado para un desarrollo rápido y sencillo. Cada día este lenguaje va ganando más adeptos, tanto así que la empresa *Sun Mirosystems*, está apoyando un proyecto llamado *Jruby* que es un intérprete de Ruby escrito 100% en Java.

Entre las características del lenguaje se encuentran:

- Posibilidad de hacer llamadas directamente al sistema operativo.
- Muy potente para el manejo de cadenas y expresiones regulares.
- No se necesita declarar las variables.
- La sintaxis es simple y consistente.
- Gestión de memoria automática.
- Todos los elementos son tratados como un objeto.
- Métodos Singleton. (10)

1.5.2 Framework

Rails es un framework para el desarrollo de aplicaciones web, software libre por naturaleza, está basado en el patrón de diseño Modelo Vista Controlador (MVC). Fue creado por David Heinemeier Hansson, empleado de la empresa 37signals.

Fue liberado por primera vez al público en julio del 2004, y lo implementó en una aplicación orientada a la administración de proyectos llamada Basecamp. Rails está basado en estos principios de desarrollo:

1. ***Don't Repeat Yourself***

La primera regla significa "No lo vuelvas a repetir", es una de los elementos más novedosos que se ha podido encontrar en este framework. Es tener un formulario, y llamarlo las veces que quieras y desde donde quieras, simplemente con una línea de código.

Tener una tabla en la base de datos, y manipular a los registros como un objeto y a sus campos como un atributo, sin necesidad de declaración, son sólo algunas aplicaciones de este principio de desarrollo.

2. ***Convention Over Configuration***

El segundo principio quiere decir "Convención antes que Configuración", con esto el *framework* le dice al programador que se ha notado que siempre se usa esto, de ésta forma. Y brinda la configuración ya realizada, si se respeta ésta configuración se ahorra tiempo, en caso contrario no hay problema. (11)

1.5.3 IDE Netbeans

El IDE NetBeans es un producto de código abierto, además, es un entorno de desarrollo integrado para los desarrolladores de software. Con su uso se obtienen todas las herramientas necesarias para crear profesionales de escritorio, de empresa, web y aplicaciones móviles con el lenguaje Java, C / C + +, e incluso lenguajes dinámicos como PHP, Java Script, Groovy, y Ruby. NetBeans IDE es fácil de instalar y utilizar, además, se ejecuta en muchas plataformas, incluyendo Windows, Linux, Mac OS X y Solaris. (12)

El IDE Netbeans 6.9 ofrece varias características o mejoras entre ellas *JavaFX Composer*, como interface gráfico para construir aplicaciones RIA mediante la tecnología *JavaFX*. *JavaFX Composer* soportará *drag&drop* de componentes y la posibilidad de hacer *binding* entre los componentes y el modelo de datos. También cuenta con soporte de la plataforma OSGI, mejoras en los editores y *debuggers* Java, Regeneración de entidades JPA ante cambios de la base de datos, Soporte de REST para Webservices y Corrector ortográfico en el editor. Además se mejora el soporte de los frameworks: JavaFX SKD 1.3, PHP Zend, Spring Framework 3.0 y Ruby on Rails 3.0. (12)

1.5.4 UML (Unified Modeling Language)

El Lenguaje de Modelación Unificado (UML) es un lenguaje gráfico para visualizar, especificar, construir y documentar. Posee formas de modelar conceptos como por ejemplo las funciones del sistema, además de otras particularidades como la de

escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusable. Usa procesos de otras metodologías, aprovechando la experiencia de sus creadores UML 03.

Entre sus características se destacan:

- Tecnología de orientación a objetos.
- Viabilidad en la corrección de errores.
- Desarrollo incremental e iterativo.
- Participación del cliente en todas las etapas del proyecto.

UML es un lenguaje consolidado, fácil de aprender, y permite una comunicación fluida entre los diversos actores acerca del modelo.

Las desventajas más notables de UML es que no ha sido diseñado para modelar procesos de negocio, por lo que no está orientado a lo que necesita el experto en el dominio del negocio, predispone un enfoque orientado a objeto lo que puede contradecir un enfoque orientado al negocio, suele estar más orientado a los arquitectos de sistemas y diseñadores de software y está pensado para un público eminentemente técnico. (13)

1.5.5 Herramienta de modelado

Visual Paradigm 3.4 for UML 6.4 Herramienta CASE que se caracteriza por soportar las últimas versiones del UML y la notación de modelado de procesos de negocio. Provee un generador de mapeo de objetos relacionales para los lenguajes de Java, .NET y PHP.

Entre las ventajas que presenta se encuentran la navegación intuitiva entre el código y el modelo, demanda en tiempo real, modelo incremental de viaje redondo y sincronización de código fuente, soporte a varios lenguajes en generación de código e ingeniería inversa a través de plataformas java, como C++, CORBA IDL, PHP, XML y Python. Brinda la posibilidad de utilizar diferentes tipos de diagramas como: diagrama de paquetes, diagrama de clases, diagrama de objetos, diagrama de componentes, diagrama de despliegue, diagrama de casos de uso y diagrama de actividades, entre otros. (14)

1.5.6 Metodología de desarrollo

Como metodología de desarrollo se seleccionó *Scrum* dada las facilidades que brinda para el desarrollo de proyectos pequeños.

"Scrum es el término que describe una forma para desarrollar productos iniciada en Japón. No se trata de un concepto nuevo, sino que ya en 1987 Ikujiro Nonaka y Hirotaka Takeuchi acuñaron este término, una estrategia utilizada en rugby en la que

todos los integrantes del equipo actúan juntos para avanzar la pelota y ganar el partido, para denominar un nuevo tipo de proceso de desarrollo de productos. Escogieron este nombre por las similitudes que consideraban que existían entre el juego del rugby y el tipo de proceso que proponían: adaptable, rápido, auto-organizable y con pocos descansos". (15)

Scrum es un proceso para la gestión y control del producto que trata de eliminar la complejidad en estas áreas para centrarse en la construcción de software que satisfaga las necesidades del negocio. Es simple y escalable, ya que no establece prácticas de ingeniería del software sino que se aplica o combina, fácilmente, con otras prácticas ingenieriles, metodologías de desarrollo o estándares ya existentes en la organización. (15)

Esta metodologia define como objetivo que los miembros del equipo trabajen juntos y de forma eficiente obteniendo productos complejos y sofisticados. Se puede entender *Scrum* como un tipo de ingeniería social que pretende conseguir la satisfacción de todos los que participan en el desarrollo, fomentando la cooperación a través de la auto-organización. De esta forma se favorece la franqueza entre el equipo y la visibilidad del producto. Se pretende que no haya problemas ocultos, asuntos u obstáculos que puedan poner en peligro el proyecto. Los equipos se guían por su conocimiento y experiencia más que por planes de proyecto formalmente definidos. El desarrollo de productos se produce de forma incremental y con un control empírico del proceso que permite la mejora continua. (15)

1.6 Conclusiones parciales

En el presente capítulo se realizó un análisis de los conceptos fundamentales asociados a la interoperabilidad, así como de las herramientas de gestión de proyecto haciendo énfasis en el Redmine. De esta herramienta se tuvo en cuentas sus principales ventajas, características y sus puntos débiles. Este análisis permitió comprender que el Redmine se puede interoperar de forma eficiente explotando varias de sus características. Por otra parte se mostraron las herramientas de planificación usadas en la UCI con las cuales se desea interoperar el Redmine, para la mejora del flujo de información. De estas se encontró como principal elemento a utilizar que exportan sus ficheros además de sus formatos convencionales en formatos estándar.

Después de todo lo analizado en el capítulo se decide construir un *plugin* para el Redmine que logre la interoperabilidad entre este y las herramientas de planificación de proyectos que son usadas en la UCI para facilitar la planificación de los proyectos.

Capítulo 2. Características y diseño del sistema

2.1 Introducción

En el presente capitulo se lleva a cabo un análisis de la propuesta de solución, así como de conceptos basados en esta, como es el caso de *plugin*. Se demuestra cómo se realizará la interacción del sistema con el redmine y el cliente a través de un mapa conceptual y un modelo de dominio. Se mencionan los requisitos funcionales y no funcionales que debe cumplir la aplicación. Además, se explican los patrones utilizados y el diagrama de clases del diseño realizado para la aplicación.

2.2 Propuesta de solución

Como propuesta de solución se presenta la creación de plugin para interoperar aplicaciones de planificación de proyectos con el Redmine.

Dicho Plugin debe cargar los artefactos generados por las aplicaciones de planificación de proyectos Microsoft Excel y Microsoft Project que exportan además de sus formatos específicos en otros estándares como son CSV(del inglés *comma-separated values*) y XML (en inglés de eXtensible Markup Language). De estos dos formatos se escogió para el trabajo el CSV ya que es un formato muy adecuado para representar de forma sencilla datos en forma de tabla, en las que las columnas están separadas por comas y las filas por retornos (lo cual se produce cuando en un editor o procesador de texto pulsamos la tecla enter/intro/entrar). Constituye una forma sencilla de representar una gran cantidad de información escrita y de que esta sea difundida, ya que al no ser un formato especialmente exigente con sistemas y tecnología, puede ser visualizada en casi cualquier entorno y mostrar el artefacto importado al Redmine luego de la elección del tipo de petición con la que se desea vincular el artefacto.

2.2.1 Concepto de plugin

Pequeño programa que añade alguna función a otro programa, habitualmente de mayor tamaño. Plugin es un programa que extiende las capacidades de un programa de un modo específico, dado por ejemplo la capacidad de mostrar vídeo, audio, ficheros de un determinado formato (ficheros PDF, presentaciones de ASAP, fichero VRML). (16)

No existe actualmente un conjunto estándar de *plugin´s* para cada tipo de ficheros, sino que existen diversas aplicaciones, realizadas por diversos fabricantes, y no todas de libre distribución. Se puede asegurar que todas las aplicaciones serán compatibles y si por ejemplo se referencia un fichero de sonido en formato .wav en su página, este

podrá ser oído por todos aquellos que tengan un *plugin* para este tipo de ficheros en su navegador.

2.2.2 Plugin del Redmine

El Redmine cuenta con una amplia librería de plugin pues esta ha sido la manera más eficiente y barata de suplir carencias que aun presenta este. Esta librería ha sido conformada en gran parte por colaboración de usuarios y desarrolladores del programa del Redmine. Ya se cuenta con más de 150 plugin en la librería del Redmine. Su creación esta en dependencia de necesidades de varios usuarios. (1)

2.3 Modelo de dominio

El Modelo de Dominio o Modelo Conceptual es una representación visual de los conceptos u objetos del mundo real, significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema. (17)

Algunas veces en dominios de negocios pequeños, no es necesario realizar un modelo de objetos para el dominio, puede ser suficiente con un glosario de términos. Por la relativa simplicidad del entorno donde se desarrolla el plugin, no es necesario profundizar a través de un modelo de negocio, basta con el modelo de dominio para capturar los principales conceptos alrededor del problema que la aplicación resuelve. (17)

2.3.1 Representación del modelo de dominio

Figura 2: Modelo de dominio

2.3.2 Entidades, conceptos y relaciones.

Para desarrollar un correcto modelo de dominio es necesario realizar una investigación de los principales conceptos utilizados en el entorno de trabajo. Por lo que a continuación se presentan algunos que son de importancia para el desarrollo del plugin.

Fichero: Es el elemento resultante de una planificación, puede ser un artefacto generado por el Microsoft Excel o por el Microsoft Project

Plugin: Es el encargado de realizar la importación hacia el Redmine.

Redmine: Esta entidad ya fue definida con anterioridad

El fichero presenta un nombre y un formato, que en este caso en específico es el CSV. Este es cargado por el plugin que es el que se encarga de interpretar los elementos del fichero, machar las tablas de contenido y modificar la tarea en caso de que esta ya se encuentre en la base de datos. El tendrá su ubicación dentro de cada proyecto del Redmine para así vincular el elemento de planificación que se importe con una tarea del proyecto.

2.4 Mapa conceptual

Los mapas conceptuales son artefactos para la organización y representación del conocimiento. Su objetivo es representar relaciones entre conceptos en forma de proposiciones. Los conceptos están incluidos en cajas o círculos, mientras que las relaciones entre ellos se explicitan mediante líneas que unen sus cajas respectivas. Las líneas, a su vez, tienen palabras asociadas que describen cuál es la naturaleza de la relación que liga los conceptos. (18)

Figura 3: Mapa conceptual

El mapa conceptual muestra cómo interactúan los conceptos, que se encuentran vinculados a la solución del problema, comenzando por Microsoft Project y Microsoft Excel, que son las herramientas de planificación que se van a interoperar. Estas exportan en el formato estándar csv, que es en el cual se debe encontrar el fichero de planificación. Este fichero contiene tareas que son las que se van a importar al plugin y luego al redmine.

2.5 Modelo del sistema

A continuación se enumeran los requerimientos del plugin y una representación del diagrama de diseño del sistema.

2.5.1 Especificación de los requerimientos del software

"Los requerimientos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requerimientos reflejan las necesidades de los clientes de un sistema que ayude a resolver algún problema como el control de un dispositivo, hacer un pedido o encontrar información. El proceso de descubrir, analizar, documentar y verificar estos servicios y restricciones se denomina Ingeniería de requerimientos". (19) A continuación se mencionan los requerimientos funcionales y no funcionales identificados.

2.5.1.1 Requerimientos funcionales

Los Requerimientos funcionales describen servicios o funciones que el sistema debe realizar.

RF1.Selecionar el fichero de planificación a importar

RF1.1 Verificar que el fichero se encuentre en formato CSV

RF1.2 Verificar el los elementos del formato del fichero

RF2. Machear las tablas

RF2.1 Verificar las tablas del fichero

RF2.2 Seleccionar las tablas de las base de datos con las cuales se va a machear las tablas del fichero

RF2.3 Seleccionar el tipo de petición

RF3.Importar las tareas que se encuentran en el fichero al Redmine

RF4.Mostrar las tareas importadas en el Redmine

2.5.1.2 Requerimientos no funcionales

Los requisitos no funcionales aseguran que se disponga de un sistema manejable y gestionable que ofrezca la funcionalidad requerida de manera fiable, ininterrumpida o con el tiempo mínimo de interrupción, incluso ante situaciones inusuales.

Requerimientos de rendimiento

El tiempo de respuesta de la aplicación debe ser bajo

El tiempo de respuesta de la pantalla debe ser bajo

La aplicación debe ser eficiente y precisa.

Requerimientos de Soporte

El plugin debe ser de fácil instalación.

Adaptable a plataformas como: Windows y Unix.

El tiempo medio de reparación debe ser menor de 24 horas.

Requerimientos de Seguridad

El plugin debe tener confidencialidad e integridad ante la información que maneja.

Solo los usuarios con permisos pueden hacer uso del plugin

Los permisos serán otorgados según la necesidad de cada usuario

Requerimientos de confiabilidad

El plugin deberá tener un 100% de disponibilidad por lo que debe estar disponible las 24 horas al día.

Todas las salidas del plugin tienen que tener el 100% de veracidad y precisión.

Requerimientos de diseño

El lenguaje de programación que se usará es Ruby on Rails.

Se requiere el uso de la librería Fastercsv.

Para la implementación del plugin se utilizará el IDE Netbeans

2.6 Patrones de arquitectura

El Redmine basa su estructura en el patrón de modelo vista controlador el cual considera dividir una aplicación en tres módulos claramente identificables y con funcionalidad bien definida: El Modelo, las Vistas y el Controlador.

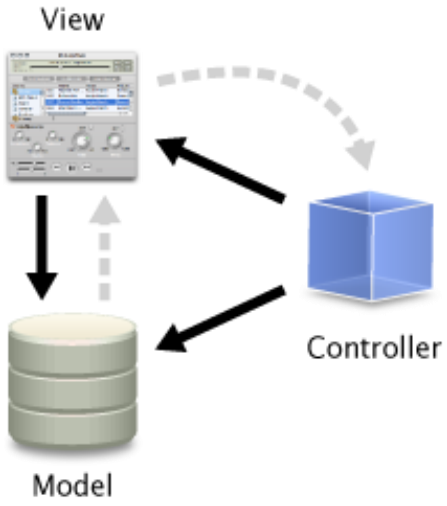


Figura 4: Modelo vista Controlador

2.6.1 El modelo

El modelo es un conjunto de clases que representan la información del mundo real que el sistema debe procesar, sin tomar en cuenta ni la forma en la que esa información va a ser mostrada ni los mecanismos que hacen que esos datos estén dentro del modelo, es decir, sin tener relación con ninguna otra entidad dentro de la aplicación.

El modelo desconoce la existencia de las vistas y del controlador. Ese enfoque suena interesante, pero en la práctica no es aplicable pues deben existir interfaces que permitan a los módulos comunicarse entre si, por lo que *SmallTalk* sugiere que el modelo en realidad este formado por dos submódulo: El modelo del dominio y el modelo de la aplicación. (20)

2.6.2 Las vistas

Las vistas son el conjunto de clases que se encargan de mostrar al usuario la información contenida en el modelo. Una vista está asociada a un modelo, pudiendo existir varias vistas asociadas al mismo modelo; así por ejemplo, se puede tener una vista mostrando la hora del sistema como un reloj analógico y otra vista mostrando la misma información como un reloj digital.

Una vista obtiene del modelo solamente la información que necesita para desplegar y se actualiza cada vez que el modelo del dominio cambia por medio de notificaciones generadas por el modelo de la aplicación. (20)

2.6.3 Controlador

El controlador es un objeto que se encarga de dirigir el flujo del control de la aplicación debido a mensajes externos, como datos introducidos por el usuario u opciones del menú seleccionadas por él. A partir de estos mensajes, el controlador se encarga de

modificar el modelo o de abrir y cerrar vistas. El controlador tiene acceso al modelo y a las vistas, pero las vistas y el modelo no conocen de la existencia del controlador. (20)

2.7 Patrones de diseño

Los patrones de diseño son descripciones de clases cuyas instancias colaboran entre sí. Cada patrón es adecuado para ser adaptado a un cierto tipo de problema. Representa un esquema o microarquitectura que supone una solución a problemas (dominios de aplicación) semejantes; una estructura común que tienen aplicaciones semejantes. (21)

Los patrones de diseño brindan una solución generalmente ya probada y documentada a problemas que se dan durante el proceso de desarrollo de software. Emplean un conjunto de buenas prácticas que facilitan el trabajo, definen una estructura de clases que da respuesta a uno o varios problemas en particular y presentan la ventaja de que son fáciles de comprender, además de que no dependen del lenguaje, haciéndolos genéricos. Lo complejo es cuando se tiene que decidir cuál usar, pues presentan diferentes soluciones, ya sea a través del empleo de uno u otro, o la combinación de varios. De ahí la importancia de conocer y estudiar los diferentes patrones que existen para poder determinar su uso.

A continuación se relacionan los patrones de diseño empleados y su aplicación en la realización de los diagramas de clases del diseño para su posterior implementación.

Patrones GRASP: Patrones que asignan responsabilidades a través de la descripción de los principios fundamentales del diseño y la solución a un problema. Entre ellos existen 5 patrones fundamentales:

- Experto: Se estableció 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 emplea este patrón para garantizar el bajo acoplamiento, encapsulación y posibilitar la reutilización de código. Las clases controladoras son las encargadas de crear las modelos y estas a su vez las entidades. Garantizando con ello la distribución por capas de la arquitectura.
- Controlador: Las clases Controladoras que son las encargadas de gestionar el acceso a lógica de negocio y controlar todo el proceso. En el 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.
- Bajo acoplamiento: Se diseñó 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, garantizando así el bajo acoplamiento.

- 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.

Patrones Gang of Four (GoF): Patrones de diseño que definen una estructura de clases que tratan una situación en particular.

- Fachada: El empleo de servicios posibilita el bajo acoplamiento y da aplicación a este patrón. El empleo de la clase interfaz de servicio permite la distribución de los servicios para la implementación de todas las funcionalidades.

2.8 Diagrama de clases del diseño

Los diagramas de clases se utilizan para modelar la vista de diseño estática de un sistema y describen su estructura. Los diagramas de clases contienen los siguientes elementos:

- Clases
- Interfaces
- Colaboraciones
- Relaciones de dependencia, generalización y asociación.

Estos diagramas son lo más importantes del diseño, contienen toda la información de las clases y sus relaciones.

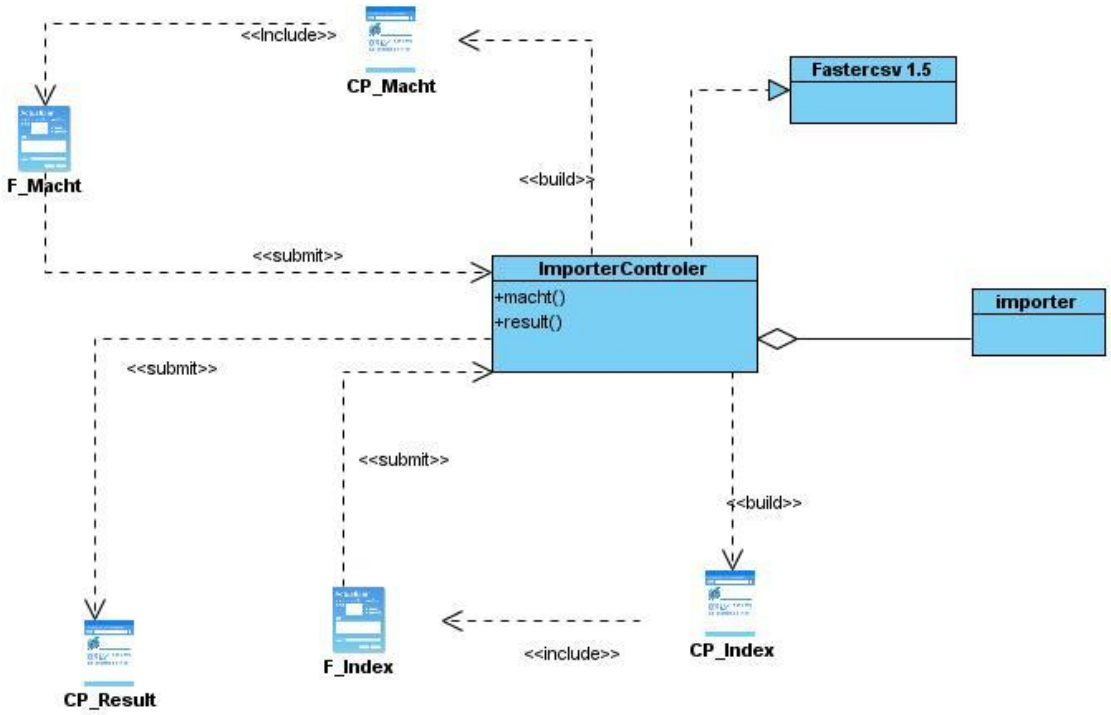


Figura 5: Diagrama de clases del diseño

Dado que el Redmine es una aplicación web, el diseño de clases se realizó basado en estereotipos web. Se muestran las clases CP_ (Client Page) que son las vistas del plugin las cuales son: CP_Index que cuenta con un formulario para subir el archivo y escoger los elementos principales del formato de este, CP_Macht esta es la clase encargada e machear las tablas del archivo y la base de datos también cuenta con un formulario para recolectar los datos que se utilizaran y seleccionar el tipo de petición con que se desea guardar en la base de datos el fichero y la clase CP_Result que es la que va mostrar el resultado de la importación. También se representan las clases Model y la clase controladora con sus dos principales métodos macht y result.

2.9 Conclusiones parciales

En el capítulo se plantean algunos conceptos necesarios para el diseño del plugin. Además se analiza en modelo de dominio y sus entidades y el modelo de diseño haciendo énfasis en los requerimientos del sistema tanto funcionales como no funcionales. Se explican los patrones de diseño utilizados, así como el diagrama de clases. Partiendo de todo lo planteado y analizado en este capítulo se logrará una eficaz implementación, pues el diseño es factor elemental para la compresión de la estructura del plugin y su composición.

Capítulo 3. Implementación y validación

3.1 Introducción

En el siguiente capítulo se mostrará lo referente a la implementación del plugin, los pasos fundamentales y las clases más importantes a la hora del trabajo. Además se exponen las pruebas realizadas al software desarrollado, con el objetivo de comprobar las funcionalidades del plugin en los diferentes escenarios, verificando en todos los casos que los resultados de las pruebas sean los esperados.

3.2 Implementación

3.2.1 Estructura del marco de trabajo

El marco de trabajo definido por la ubicación del Redmine como principal componente de trabajo, dentro del cual se realizarán todas las modificaciones vinculadas a la creación del plugin basado siempre en una arquitectura web. Se presenta la carpeta raíz del Redmine y los componentes que se encuentran dentro de la misma.

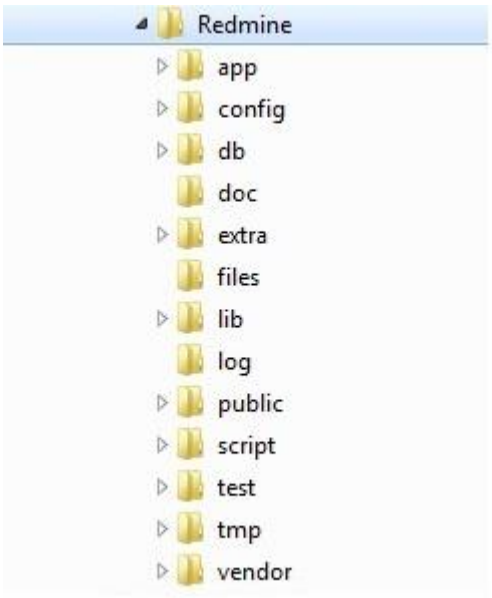


Figura 6: Estructura del marco de trabajo Redmine

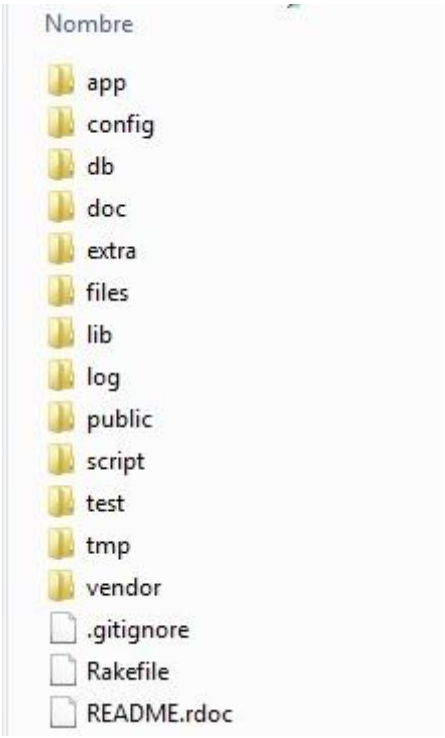


Figura 7: Componentes del las raíz Redmine

Dentro del directorio Redmine la especificación de las configuraciones se trabajara en la carpeta *config* en ella se ven los elementos configurables como la basedatos y los idiomas. Se centra la explicación del marco de trabajo en la carpeta **vendor** pues es en ella en la que se va a trabajar en la creación del plugin.

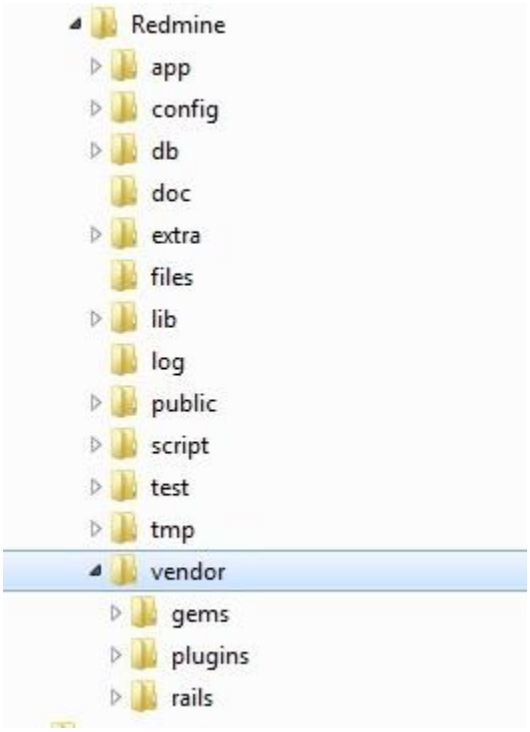


Figura 8: Marco de trabajo vendor

Dentro de la carpeta **vendor** se encuentran los componentes: **gems** que es donde se encuentran algunas de las librerías que en el caso del Redmine se denominan gemas, rails componente vinculado al frameworks y *plugins* que es donde se crean y almacenan los plugin del Redmine.

Figura 9: Macro de trabajo plugins

El plugin desarrollado lleva por nombre **redmine_importer** dentro del cual se encuentran los elementos creados para el funcionamiento del mismo.

Figura 10: Marco de trabajo plugin redmine_importer

El directorio **app** es en el cual se encuentran las principales clases del plugin ordenadas según su funcionalidad o características.

Figura 11: Marco de trabajo app

- **Controllers**: Contiene las clases controladoras correspondientes. Está encargada de gestionar las funcionalidades del componente y el flujo de información entre las vistas y las modelos. Comunica las interfaces de usuario con la lógica del negocio.
- **Model**: Contiene la programación de toda la lógica del negocio del componente así como el control y acceso a la información física de la base de datos.
- **Views:** Contiene los ficheros que gestionan la capa de presentación, agrupados en la carpeta **importer.**

3.2.2 Estándares de codificación

Los estándares de codificación son pautas de programación que están enfocadas a la estructura de la lógica del código para facilitar su lectura, comprensión y mantenimiento.

Las normas de codificación definidas por la línea de arquitectura están enfocadas a lograr una mejor comprensión del código que responda a la complejidad dada por la cantidad de componentes y el alto nivel de integración que existe entre ellos. Esto permite un mejor entendimiento por parte de la gran cantidad de personal involucrado en el desarrollo del sistema, además de garantizar un código legible y reutilizable.

3.2.2.1 PascalCasing

PascalCasing establece que los nombres de los identificadores, las variables, métodos y clases están compuestos por una o más palabras juntas, iniciando cada palabra con letra mayúscula y el resto en minúscula.

Nomenclatura de las clases según su tipo.

Todas las clases están nombradas siguiendo el estándar PascalCasing, nombrándolas de acuerdo al propósito y la función de la misma.

- Controllers: Constituyen las clases controladoras del dominio y su identificador está estructurado por el nombre de la misma seguido por la palabra “Controller”. Ejemplo: ImporterController.

- Bussines: Constituyen las clases que gestionan la lógica del negocio y su identificador está estructurado por la función que realizan seguido de la palabra “Model”. Ejemplo: ImporterModel.
- Domain: Constituyen las clases que gestionan el acceso a la base de datos a través de consultas y su identificador está estructurado por el nombre de la tabla a la que acceden, pueden incluir el prefijo “Dat” o “Nom”. Ejemplo: DatImporter y NomImporter.
- Generated: Constituyen las clases bases del dominio encargadas de realizar el mapeo de modelo de objetos al modelo relacional, su identificador está estructurado por el prefijo “Base”, seguido de “Dat” o “Nom” y luego el nombre de la tabla a la que hacen referencial. Ejemplo: BaseDatImporter y BaseNomImporter.
- Validators: Constituyen las clases validadoras y su identificador está estructurado por el identificador seguido de la palabra “Validator”. Ejemplo: ImporterValidator.
- Services: Constituyen las clases que ofrecen servicios en los componentes, su identificador está estructurado con un nombre sugerente de acuerdo a las operaciones que realizan y prestaciones que brindan, seguido de la palabra “Service”. Ejemplo: ImporterService.

3.2.2.2 CamelCasing

CamelCasing es similar a PascalCasing con diferencia en la letra inicial del identificador que no comienza con mayúscula. Esta notación se utilizó para el nombre de funciones y atributos.

Nomenclatura de las funciones.

El identificativo de todas las funciones o métodos se escribe con la primera palabra en minúscula de acuerdo a la función que realizan. Ejemplo:

ObtenerOperacionesGuardadas.

Los denominadores de las acciones de las clases controladoras tienen la peculiaridad de ir seguidos por la palabra “Action”. Ejemplo: importAction.

Nomenclatura de los atributos.

El identificativo de los atributos se escribe de acuerdo a su objetivo con la primera letra en minúscula. Ejemplo: idIsues y newIsues.

3.2.2.3 Notación húngara

Notación también conocida como REDDICK por el nombre de su autor está basada en definir prefijos para cada tipo de datos según el ámbito de las variables, con la finalidad de lograr mayor comprensión del nombre de la variable, método o función.
Esta notación fue utilizada de acuerdo a los siguientes prefijos:

Tipo de Datos	Prefijo
Arreglos	arr
Objetos	obj
Enteros	int
Cadenas	cad
Float	flt
Boolean	bool

Tabla 1: Tipo de datos

Nomenclatura de las variables.

El identificativo de los atributos se escribe atendiendo a su objetivo y de acuerdo al estándar CamelCasing, con la primera letra en minúscula. Ejemplo: arrCuenta, que define un arreglo de cuentas.

Nomenclatura de las constantes.

El identificativo de las constantes se realiza utilizando todas las letras en mayúscula. Ejemplo: SUBSISTEMA.

3.2.3 Descripción de clases

Se identifican las clases fundamentales y las funcionalidades de estas. Se escoge como eje central del enfoque de este epígrafe la clase controladora pues es la más importante en el desarrollo del plugin. Las cases**: índex, macht y result** son vistas por lo que su descripción no es significativa.

Clase controladora

Nombre: ImporterController	
Tipo de clase: Controladora	
Para cada responsabilidad	
Funcionalidad	**Descripción**

machtAction	Machear las tablas o campos de la petición entrada con los de la base de datos
machtAction	Insertar el elemento en base de datos y mostrar los resultados

Tabla 2: Descripción de la clase ImporterController

En la clase ImporterControler es en donde se van a realizar todas las funcionalidades del plugin distribuidas en dos métodos fundamentales machtAction y machtAction, en el primero se toman las tablas que contiene el fichero que se desea importar y se verifica si alguno de estos coincide con los campos ya determinados por el Redmine, además se verifica el formato del fichero de entrada. En el segundo ya establece la conexión a la base de datos y se importa en fichero mostrando luego los resultados.

3.3 Validación

En el desarrollo de cualquier sistema a medida que este avanza se van introduciendo errores y aun cuando se ha finalizado la creación se pueden encontrar no conformidades en el mismo. Es imposible asegurar que un software se encuentre completamente libre de errores; sin embargo, existen formas y métodos para acercarse lo más posible a un resultado óptimo, o el esperado por el cliente.

3.3.1 Métricas para la evaluación del modelo de diseño.

Para realizar la evaluación del modelo de diseño del plugin se aplicaron diferentes métricas, de las cuales a continuación se describen y muestran los resultados.

3.3.1.1 Métrica de Tamaño Operacional de Clase (TOC).

La métrica TOC está dada por la cantidad de funcionalidades contenidas en las clases, a partir de las cuales se determina la afectación que ejerce en el diseño.

Tamaño Operacional de Clase (TOC)	
Atributos	**Afectación**
Responsabilidad	El aumento del TOC provoca un aumento de la responsabilidad asignada a la clase.
Complejidad de Implementación	El aumento del TOC provoca un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC provoca una disminución en el grado de reutilización

	de la clase.

Tabla 3: Afectaciones en el diseño según la métrica TOC.

Para la evaluación de cada atributo se establece un rango de valores que determinan la complejidad en la aplicación del TOC. Estos se muestran en la siguiente tabla.

	Categoría	Criterio
Responsabilidad	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.
Complejidad implementación	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.
Reutilización	Baja	> 2*Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	<= Prom.

Tabla 4: Rango de valores de para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC.

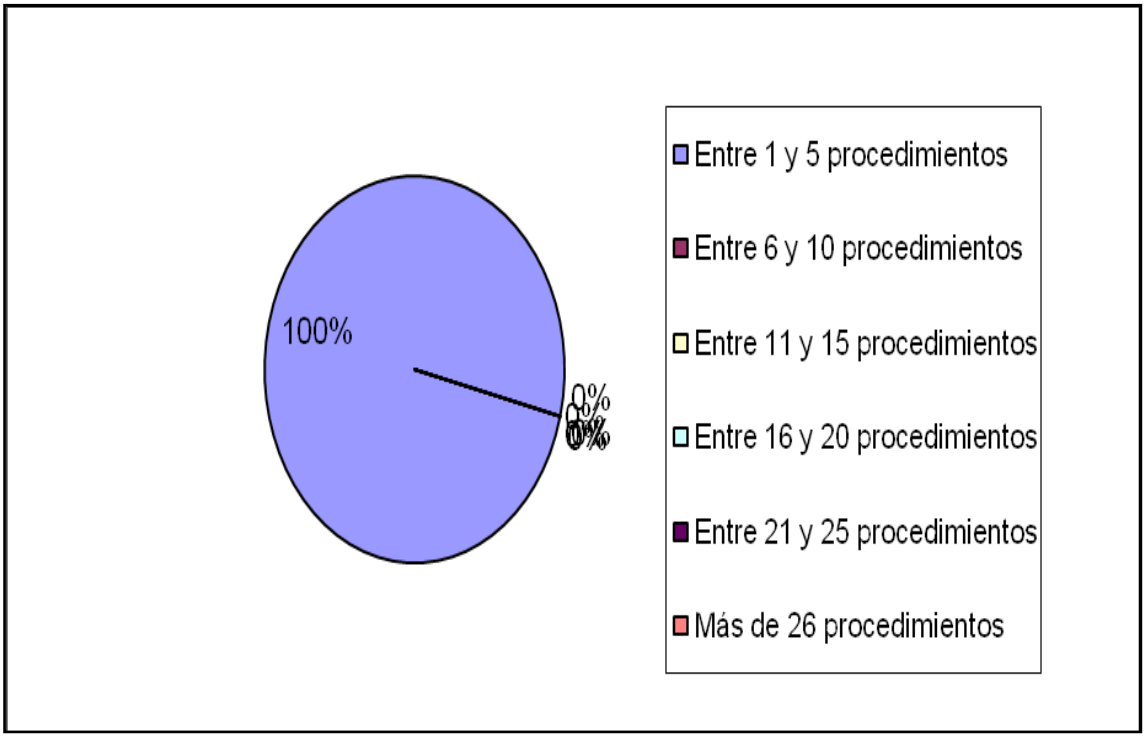


Figura 12: Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos según la métrica TOC.

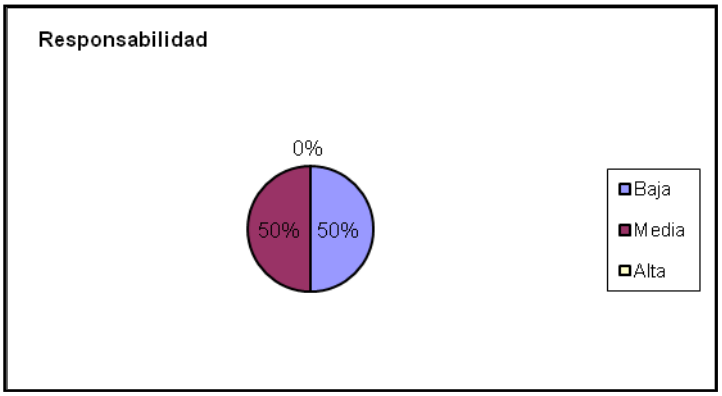


Figura 13: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.

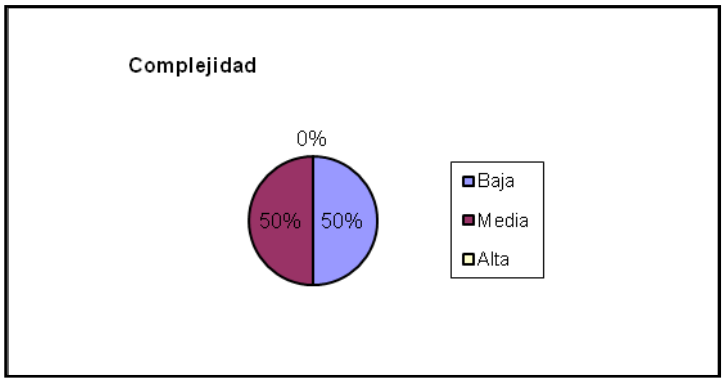


Figura 14: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación.

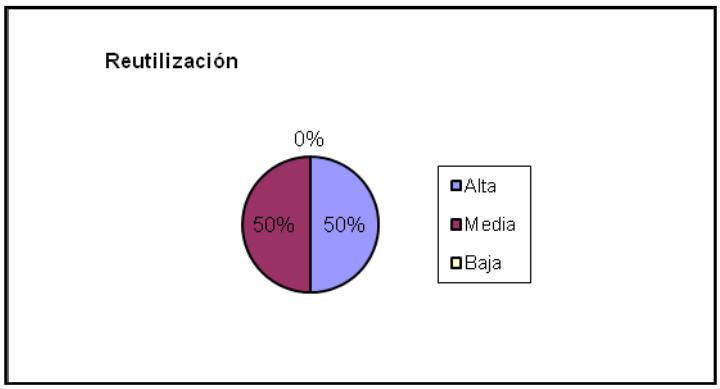


Figura 15: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización

Durante la evaluación de la métrica TOC los resultados obtenidos demostraron que la mayoría de las clases se encuentra dentro de los niveles aceptables de calidad. Los atributos de calidad de las clases se encuentran en un nivel medio satisfactorio (50%); de manera que se puede confirmar la elevada reutilización (elemento clave en el proceso de desarrollo de software) y cómo se reducen en menor grado la responsabilidad y la complejidad de implementación.

3.3.1.2 Métrica de Relaciones entre Clases (RC).

La métrica RC está dada por la cantidad de relaciones de uso existentes entre las clases contenidas en el diseño, a partir de las cuales se determina la afectación que ejerce.

Relaciones entre Clases (RC)	
Atributos	**Afectación**

Acoplamiento	El aumento del RC provoca un aumento del acoplamiento de la clase.
Complejidad del mantenimiento	El aumento del RC provoca un aumento de la complejidad del mantenimiento de la clase.
Reutilización	El aumento del RC provoca una disminución en el grado de reutilización de la clase.
Cantidad de pruebas	El aumento del RC provoca un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 5: Afectaciones en el diseño según la métrica RC.

Para la evaluación de cada atributo se establece un rango de valores que determinan la complejidad en la aplicación de la RC.

A continuación se muestra el rango de valores y la complejidad por cada uno de los atributos.

	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2

	Categoría	Criterio
Complejidad Mant.	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

	Categoría	Criterio
Reutilización	Baja	>2* Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	<= Prom.

	Categoría	Criterio

Cantidad de Pruebas	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

Tabla 6: Rango de valores de para la evaluación técnica de los atributos de calidad relacionados con la métrica RC.

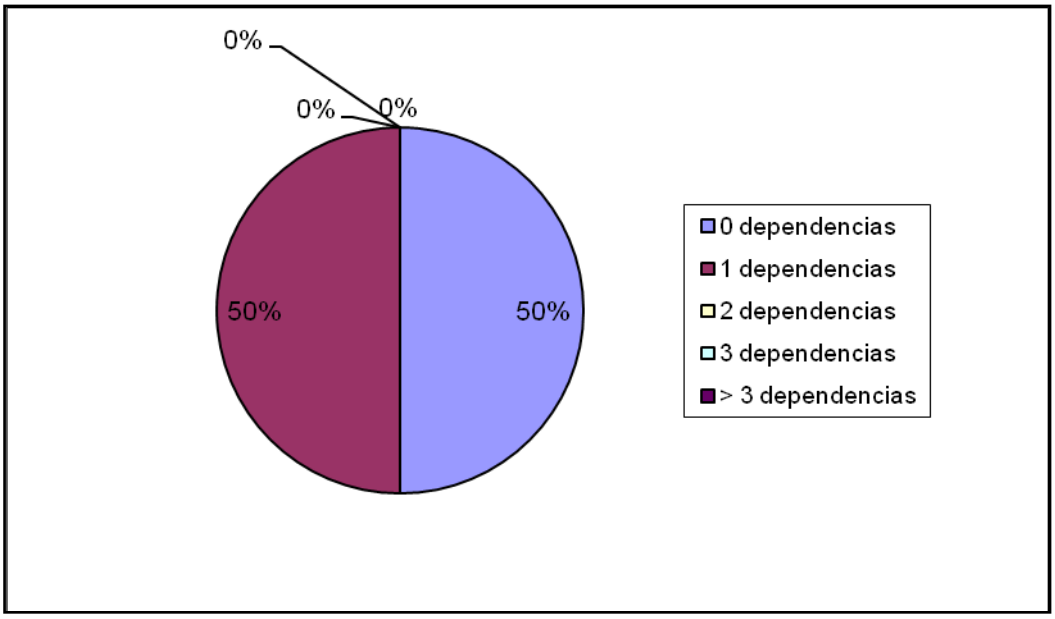


Figura 16: Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos según la métrica RC.

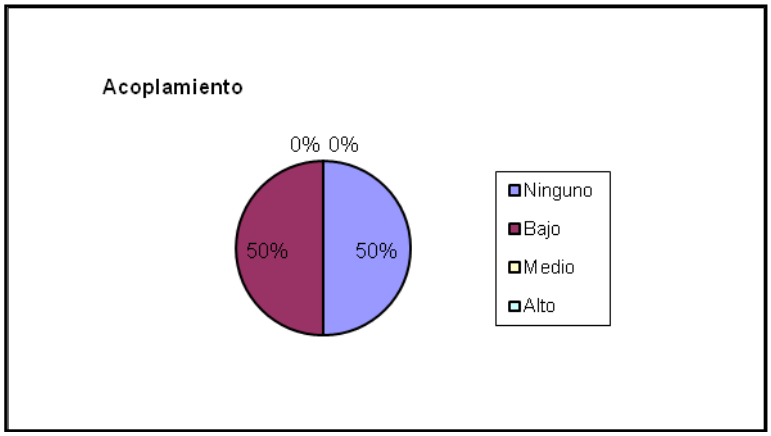


Figura 17: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento.

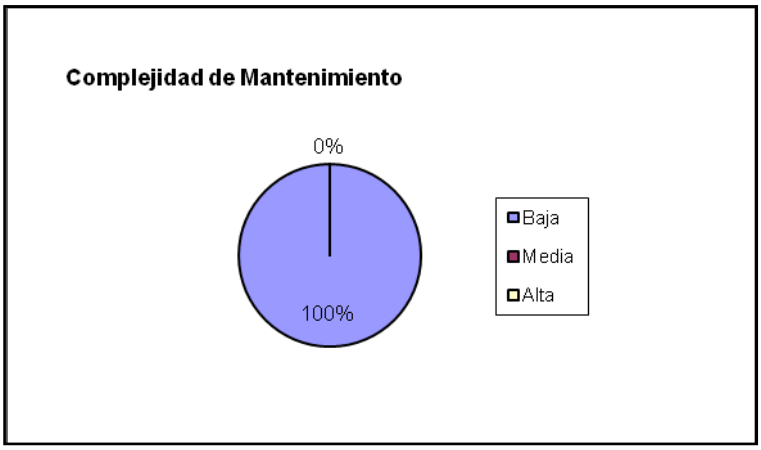


Figura 18: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento.

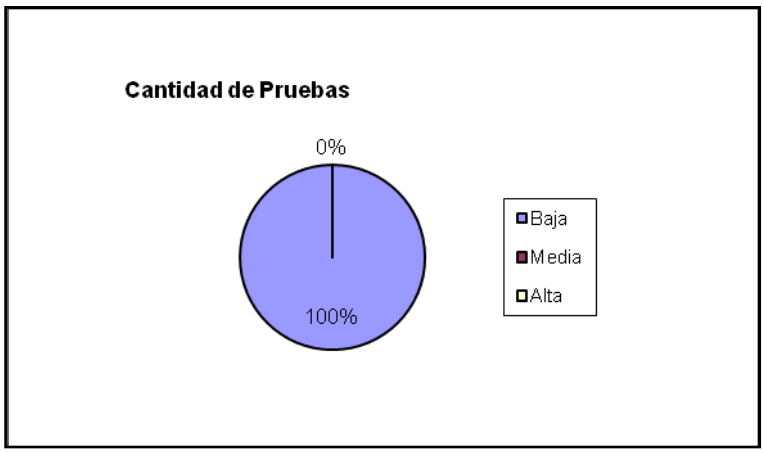


Figura 19: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas

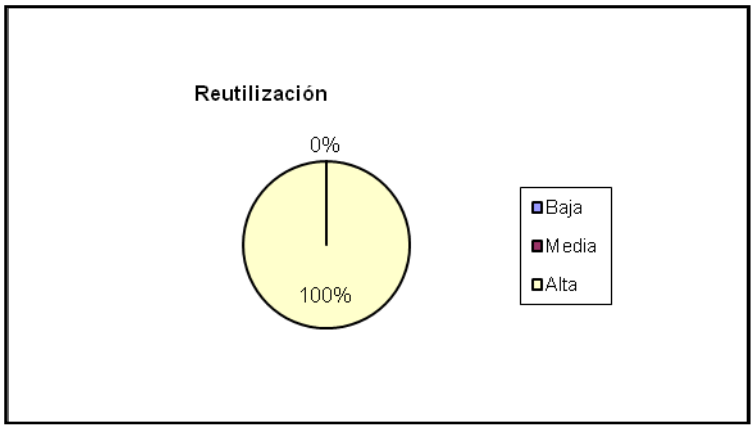


Figura 20: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización.

Durante la evaluación de la métrica RC los resultados obtenidos demostraron que la mayoría de las clases (100%), poseen menos de 3 dependencias entre clases, por lo que se encuentra dentro de los niveles aceptables de calidad. Los atributos de calidad se encuentran en un nivel satisfactorio; en el 50% de las clases el grado de dependencia o acoplamiento es mínimo, la complejidad de mantenimiento, la cantidad de pruebas y la reutilización se comportan favorablemente para un 100% de afectación en las clases.

3.3.2 Pruebas

Las pruebas tienen gran importancia en el desarrollo de un software, ya que mediante éstas se pueden detectar y corregir errores tempranamente. Antes de entregar el producto al cliente final se debe garantizar que el software cumpla con todos los requerimientos y se han corregido todos los errores, pues cada vez que el programa se ejecuta, el cliente lo está probando, por tanto, debemos hacer un intento especial para que se sienta satisfecho. Con el objetivo de encontrar el mayor número posible de errores, las pruebas deben conducirse sistemáticamente y los casos de prueba deben diseñarse utilizando técnicas definidas.

3.3.2.1 Niveles de prueba aplicados.

Se definieron diferentes tipos de pruebas aplicables a diferentes niveles para el correcto funcionamiento del *plugin*. Las pruebas aplicadas son las siguientes:

- Pruebas Internas: Diseñadas e implementadas por el grupo de calidad interna para verificar el correcto funcionamiento del *plugin*.
- Pruebas de Integración: Se verifica que el *plugin* opere correctamente cuando se combine con el Redmine.

Estrategia de pruebas.

Se define una estrategia de pruebas regida según el modelo de desarrollo establecido, con el fin de garantizar la calidad del sistema.

Se especifican los casos de pruebas, a partir de la construcción de todos los posibles caminos de ejecución, o escenarios, basados en los requisitos implementados, comparando cada funcionalidad descrita con lo implementado para verificar hasta que punto concuerda y cumple con las necesidades del cliente. Se obtiene como resultado un listado final con los casos de prueba identificados a partir de los posibles escenarios, los resultados esperados para

cada caso y las condiciones o valores requeridos para la ejecución de los distintos escenarios.

Se ejecutan pruebas internas del sistema como la de caja negra, antes de incorporarlo al equipo central de calidad para que realice la pruebas de liberación, tratando de garantizar las detecciones de la menor cantidad de no conformidades por este.

Se realizan además tantas iteraciones como sean necesarias para garantizar la calidad del sistema a desplegar y comprobar que el software funciona según las expectativas del cliente, y que se encuentra ejecutando las funciones y tareas para las cuales fue construido.

3.3.2.2 Diseño de casos de prueba.

Caja negra

Los casos de prueba para caja negra se centran en realizar pruebas de software para comprobar su funcionalidad. Se encuentran basados en diferentes entradas que puede recibir el sistema con sus correspondientes valores de salida.

Los casos de prueba demuestran que:

- Las funciones del software son operativas.
- Las entradas se aceptan de la forma adecuada produciendo el resultado esperado.
- La integridad de la información externa (por ejemplo archivos de datos) se mantiene.

Para la verificación de los requisitos definidos en el diseño, se esbozan los casos de prueba y se realizan las pruebas internas a la aplicación.

A continuación se especifica el caso de prueba “Importar el elemento al Redmine”

Condiciones de ejecución.

- Se debe identificar y autenticar ante el sistema y además debe tener los permisos para ejecutar esta acción.
- Se debe acceder a la pestaña importar
- Se debe escoger el elemento a importar y comprobar la configuración del archivo csv.

- Se debe machear las tablas con las de la base de datos y escoger el tipo de petición.

Requisito a probar.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Importar el elemento al Redmine	Importar el elemento al Redmine	EP 1.1: Importar el elemento al Redmine con datos validos.	– Se presiona el botón **Submit** – Se inserta el elemento en la tabla en la base de datos del Redmine
		EP 1.2: Importar el elemento al Redmine con datos no validos.	– Se presiona el botón **Submit** – No se inserta el elemento en la tabla en la base de datos del Redmine y se notifica el error
		EP 1.3: Importar el elemento al Redmine con datos incompletos.	– Se presiona el botón **Submit** – No se inserta el elemento en la tabla en la base de datos del Redmine y se notifica el error
		EP 1.4: Cancelar la operación.	– Se presiona el botón **Adicionar**. – Se inserta o no el elemento en la tabla en la base de datos del Redmine.

			– Se presiona cualquier botón.

Tabla 7: Descripción del caso de prueba para el requisito Importar el elemento al Redmine

Descripción de variable.

No.	Nombre de campo	Clasificación	Válido	Inválido
1	Petición	Petición	NA	NA

Tabla 8: Descripción de las variables del caso de prueba para el requisito Importar el elemento al Redmine

3.4 Resultados de las pruebas

Para los diferentes niveles de pruebas definidos se obtuvieron resultados que se encuentran dentro de los parámetros de aceptación. Validando la implantación llevada a cabo.

En el caso de las pruebas internas, estas se realizaron a través de las pruebas de caja negra, comprobando la implementación de todos los requisitos funcionales. A través de los *Sprint*[3] realizados se obtuvieron un número de no conformidades las cuales fueron corregidas. A continuación se muestra una tabla con los valores obtenidos.

[3] Sprint: En la metodología *Scrum* es el término que denomina a una iteración que esta acotada en el tiempo usualmente entre dos y cuatro semanas.

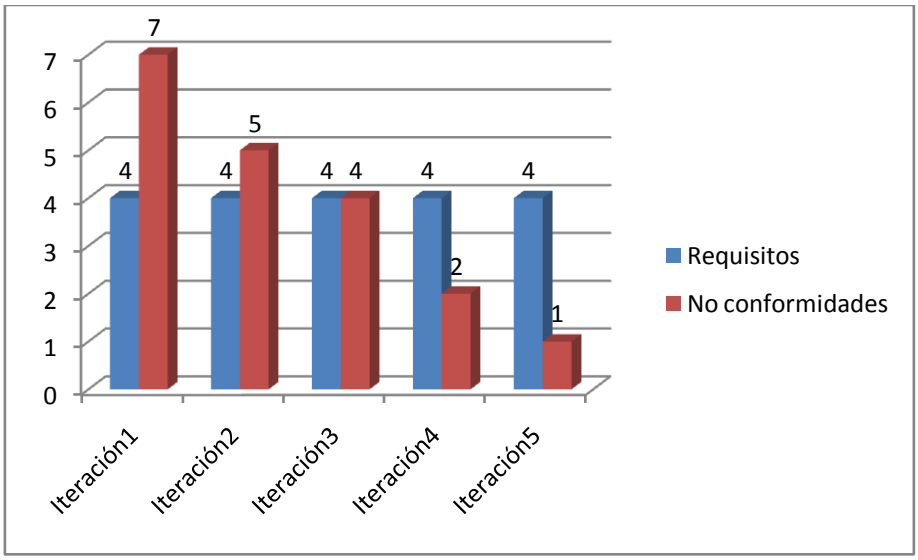


Figura 21: No conformidades por iteración

En las pruebas de integración no se obtuvo ninguna no conformidad demostrando la adaptación del plugin al entorno del redmine de manera satisfactoria.

3.5 Conclusiones parciales

Con la realización y análisis del siguiente capítulo se comprueba como con la implementación se mantuvo la flexibilidad arquitectónica y la solidez del funcionamiento del plugin, y se muestra además el marco en cual se encuentra el mismo.

Por la validación se arribó a que; el diseño desarrollado cumple con los diferentes aspectos medidos a partir de la caracterización cuantitativa obtenida de la evaluación a través de la aplicación de las métricas de diseño. Además se realizaron pruebas de funcionamiento que permitieron evaluar los diferentes elementos de la aplicación y comprobar que las funcionalidades implementadas las cuales mostraron un correcto funcionamiento y una respuesta a los requisitos funcionales que garantizan las necesidades de los clientes.

Por tanto se concluye que en este capítulo se implementó y se comprobó la obtención de un plugin en estado funcional.

Conclusiones

Con la realización del siguiente trabajo se logró resolver los problemas de interoperabilidad que existían entre el Redmine y otras aplicaciones de planificación de proyectos. Se logró la creación de un plugin que cumple con las normas de diseño e implementación y que además abarca todas las funcionalidades requeridas por el cliente.

Para logar este resultado final se cumplieron los objetivos específicos trazados:

- Se lograron Identificar las posibles herramientas de planificación de proyectos para la comunicación las cuales fueron Microsoft Excel y Microsoft Project y las posibilidades de interoperabilidad que brinda el Redmine. permitiendo utilizar las ventajas que brindan dichas herramientas para la creación del plugin.
- Se definió el marco teórico de la investigación el cual ayudo a la compresión y desarrollo del sistema así como la elección de las herramientas usadas en el diseño e implementación.
- Se diseñó e implementó un plugin para importar tareas al Redmine. logrando así la interoperabilidad entre las herramientas mencionadas anteriormente y este.
- Se evaluaron los resultados obtenidos. permitiendo mejorar la calidad del producto final

Dando cumplimiento además a todas las tareas de investigación propuestas.

La investigación realizada y el trabajo llevado a cabo proporcionan al Redmine de una funcionalidad de la cual carecía.

Recomendaciones

A partir del estudio realizado se recomienda:

- La utilización del plugin y que se extienda su alcance
- Continuar con la realización de pruebas funcionales que permitan garantizar la calidad del producto.
- Internacionalizar el plugin para su uso de toda la comunidad del Redmine.

Referencia Bibliográfica

1. **Centro de excelencia de software libre de Castilla La Mancha.** *Analisis de la aplicacion: Redmine.* 2010.

2. *The Interoperability, a Desert Storm case study.* **Sterling D. Sessions, Carl R Jones.** [ed.] National Defense University. Washington D.C : s.n., 1993.

3. *E-Government Act of 2002.* 2002. Vol. US H.R. 2458.

4. **CE del Parlamento Europeo y del Consejo.** *Decisión 2004/387/. Prestación interoperable de servicios paneuropeos de administración electrónica al sector público, las empresas y los ciudadanos (IDABC).* 2004.

5. *INTEROPERABILIDAD; ESTÁNDARES.* **Castañeda, Ing. Luz María.** 2004, Revista Digital Universitaria, Vol. 5. ISSN: 1067-6079..

6. *DA. European Interoperability Framework for pan-European egovernment services: Framework. IDA working document, version 4.2. European Communities.* **Marco Europeo de Interoperabilidad (MEI).** 2004.

7. **Claudia Soledad Soria, Martín Roberto Tolava.** *Redmine, la aplicación Web 2.0 para la Gestión de Proyectos.* s.l. : Universidad Tecnológica Nacional – Facultad Regional Tucumán – Dirección de Proyectos , 2009.

8. **Microsoft.** Microsoft. office.microsoft.com. [En línea] Microsoft, 2011. http://office.microsoft.com/es-es/..

9. **Universidad Tecnológica del Centro.** Unitec. [En línea] Equipo web Unitec, 2011. webteam@unitec.edu.ve.

10. **Ruby-lang.org.** Ruby. *Ruby.* [En línea] 2011. http://www.ruby-lang.org/es/.

11. **rubyonrails.org.** Ruby on Rails. [En línea] 2011. http://rubyonrails.org/..

12. **Cerda, Felipe.** NetBeans 6.5 El único IDE que necesitas. [En línea] 2010. ht t p: / / bl ogs . s un. com.

13. **OMG.** uml.org. *UML® Resource Page.* [En línea] http://www.uml.org/.

14. **Visual Paradigm.** visual-paradigm. *visual-paradigm.* [En línea] http://www.visual-paradigm.com/.

15. **González, Pilar Rodríguez.** *Estudio de la aplicación de metodologías ágiles para la evolución de productos software* . 2010.

16. **definiciones.org.** Definiciones. [En línea] http://www.definicion.org/plug-in.

17. **Dapena, MSc. Martha D. Delgado.** *Definición del modelo del negocio y del dominio utilizando Razonamiento Basado en Casos.* s.l. : Centro de Estudios de Ingeniería de Sistemas., 2010.

18. *Mapas Conceptuales.* **Dürsteler, Juan C.** s.l. : La revista digital de InfoVis.net, 2004.

19. **Sommerville, Ian.** *Ingenieria del software.* . Septima edición. Madrid : Pearson educacion, S.A. Madrid, 2005.

20. **Pantoja, Ernesto Bascón.** *El patrón de diseño Modelo-Vista-Controlador (MVC)* . 2010.

21. **Ramiro Lago.** Patrones de diseño software. [En línea] abril de 2007. http://www.proactiva-calidad.com/java/patrones/.

22. **Mestras, Juan Pavón.** *Estructura de las Aplicaciones Orientadas a Objetos,El patrón Modelo-Vista-Controlador (MVC).* s.l. : Dep. Ingeniería del Software e Inteligencia Artificial Universidad Complutancion Madrid, 2009.

23. **Lago, Ramiro.** Patrones de diseño software. [En línea] Abril de 2007. http://www.proactiva-calidad.com/java/patrones/.

Bibliografía

Brad Ediger *Advances Rails,* Published by O'Reilly Media, Inc, 2008

Castañeda, Ing. Luz María *INTEROPERABILIDAD; ESTÁNDARES.***.** 2004, Revista Digital Universitaria, Vol. 5. ISSN: 1067-6079..

CE del Parlamento Europeo y del Consejo. *Decisión 2004/387/. Prestación interoperable de servicios paneuropeos de administración electrónica al sector público, las empresas y los ciudadanos (IDABC).* 2004.

Centro de excelencia de software libre de Castilla La Mancha. *Analisis de la aplicacion: Redmine.* 2010.

Cerda, Felipe. NetBeans 6.5 El único IDE que necesitas. [Online] 2010. ht t p: / / bl ogs . s un. com.

Dan Chak *Enterprise Rails,* Published by O'Reilly Media, Inc ,2009 [online] http://www. *Enterpriserails.chak.org*

Dapena, MSc. Martha D. Delgado. *Definición del modelo del negocio y del dominio utilizando Razonamiento Basado en Casos.* s.l. : Centro de Estudios de Ingeniería de Sistemas., 2010.

definiciones.org. Definiciones. [Online] http://www.definicion.org/plug-in.

E-Government Act of 2002. 2002. Vol. US H.R. 2458.

González, Pilar Rodríguez. *Estudio de la aplicación de metodologías ágiles para la evolución de productos software .* 2010.

Marco Europeo de Interoperabilidad (MEI)*DA. European Interoperability Framework for pan-European egovernment services: Framework. IDA working document, version 4.2. European Communities.***.** 2004.

Mestras, Juan Pavón. *Estructura de las Aplicaciones Orientadas a Objetos,El patrón Modelo-Vista-Controlador (MVC).* s.l. : Dep. Ingeniería del Software e Inteligencia Artificial Universidad Complutancion Madrid, 2009.

Microsoft. Microsoft. office.microsoft.com. [Online] Microsoft, 2011. http://office.microsoft.com/es-es/..

OMG. uml.org. *UML® Resource Page.* [Online] http://www.uml.org/.

Pantoja, Ernesto Bascón. *El patrón de diseño Modelo-Vista-Controlador (MVC) .* 2010.

Ruby-lang.org. Ruby. *Ruby.* [Online] 2011. http://www.ruby-lang.org/es/.

rubyonrails.org. Ruby on Rails. [Online] 2011. http://rubyonrails.org/..

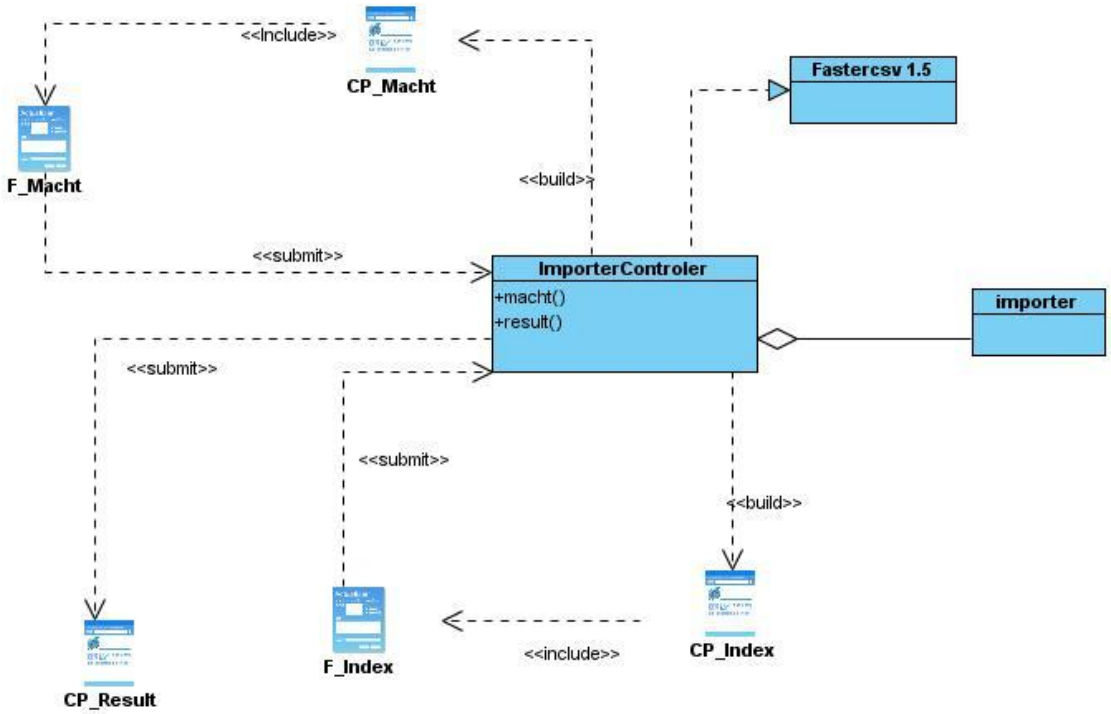
Sam Ruby, Dave Thomas y David Heinemeier Hansson *Agile Web Development with Rails* 2009

Sterling D. Sessions, Carl R Jones *The Interoperability, a Desert Storm case study.* [ed.] National Defense University. Washington D.C : s.n., 1993.

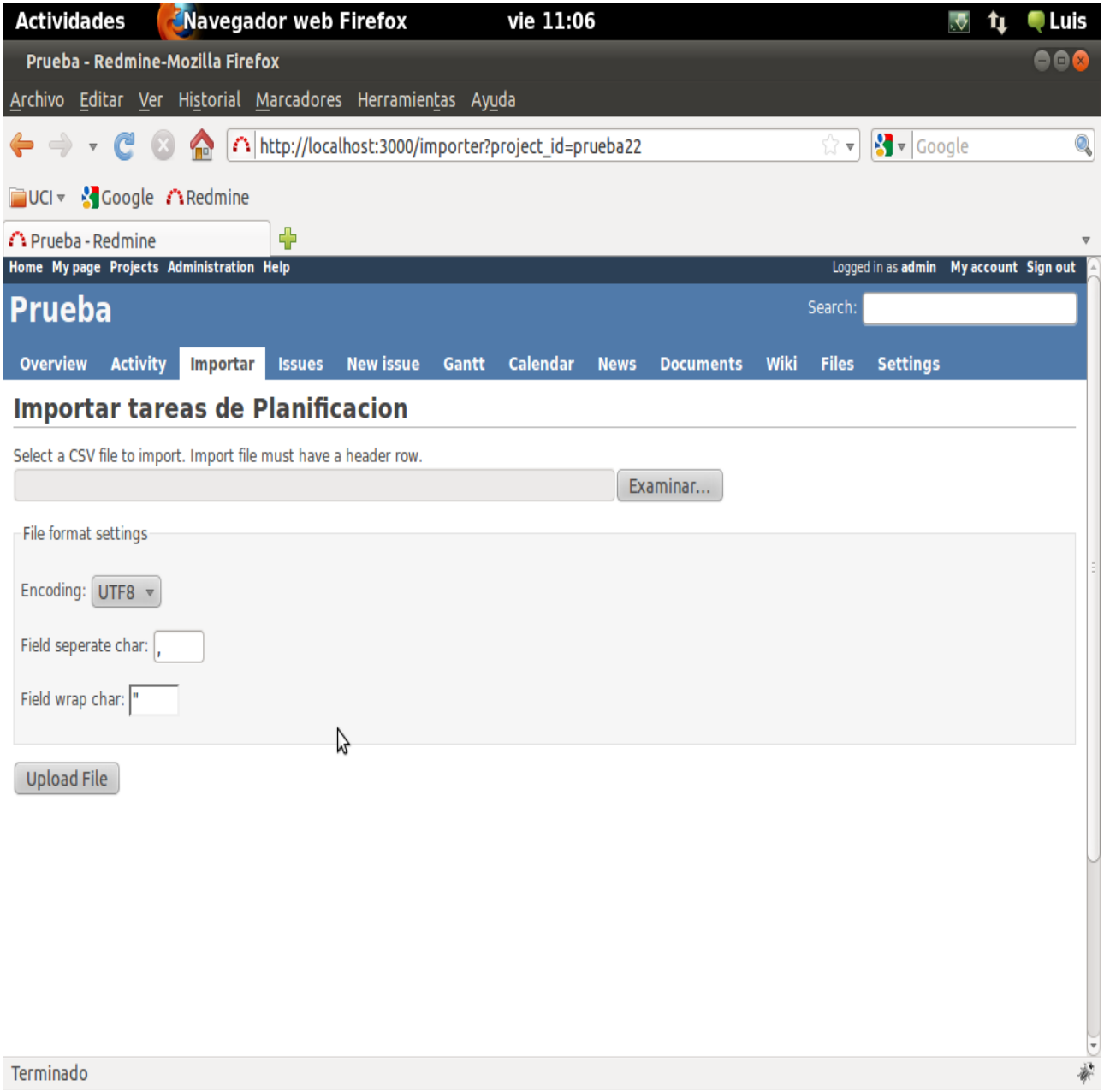
Universidad Tecnológica del Centro. Unitec. [Online] Equipo web Unitec, 2011. webteam@unitec.edu.ve.

Visual Paradigm. visual-paradigm. *visual-paradigm.* [Online] http://www.visual-paradigm.com/.

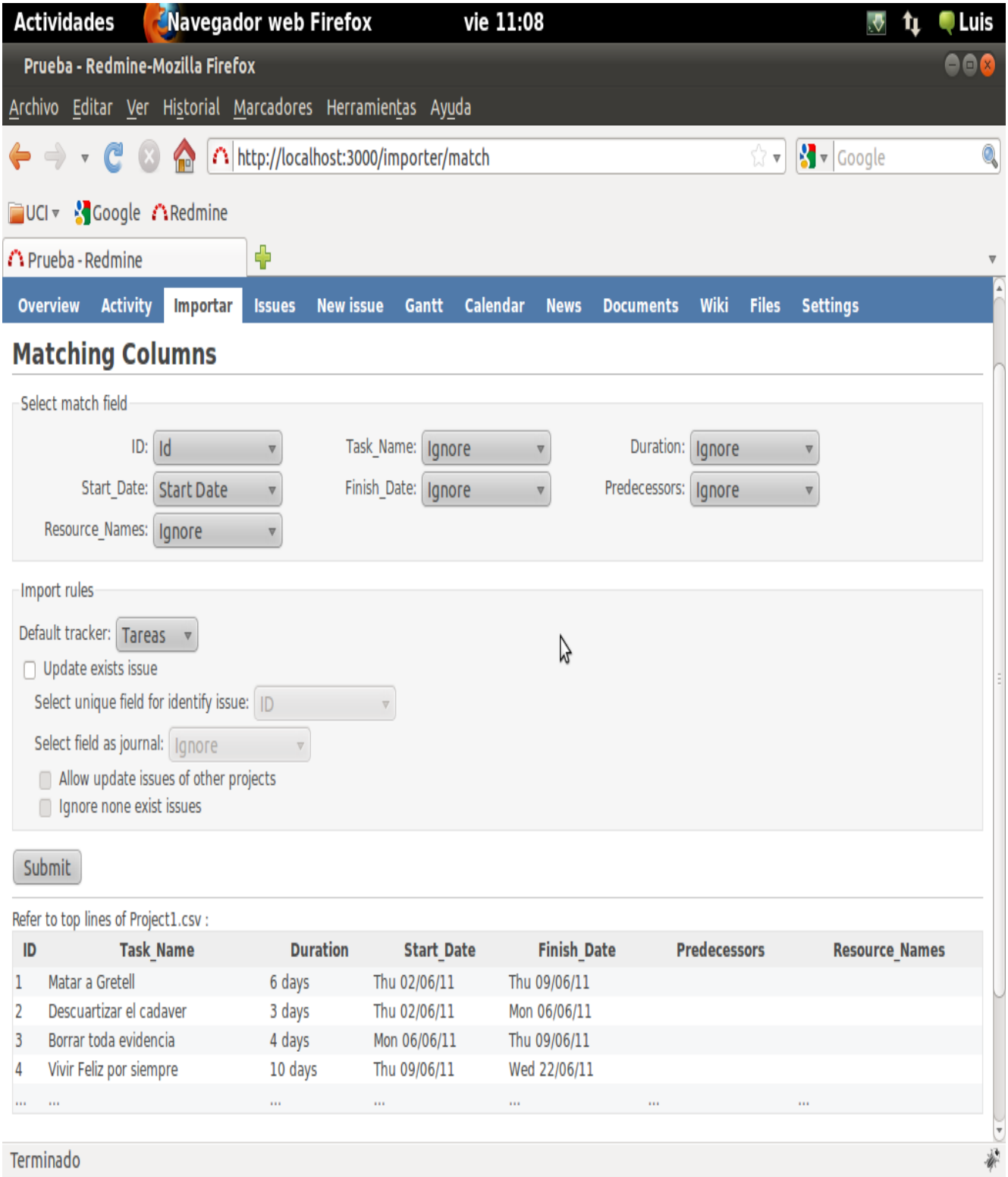
Anexos



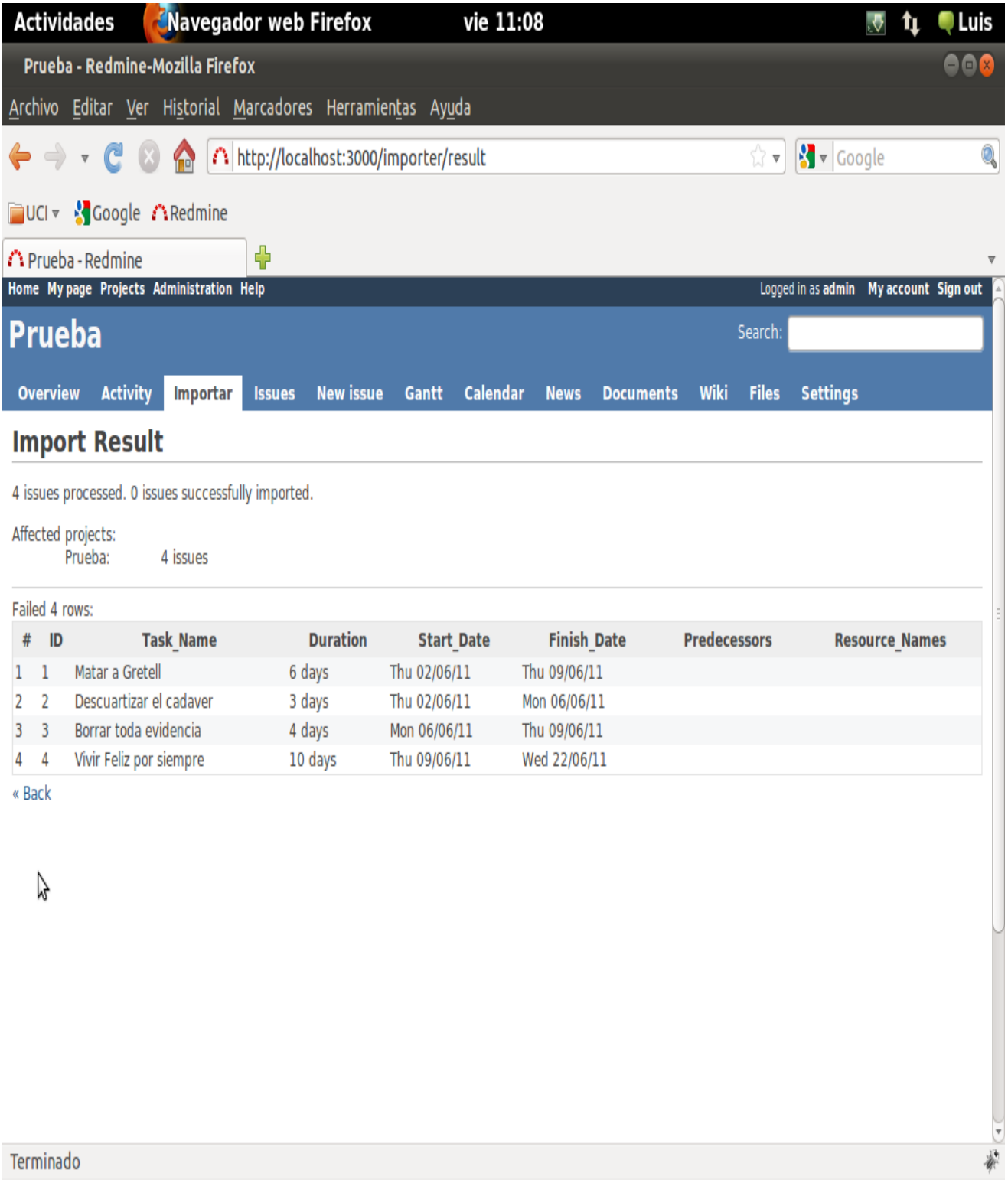
Anexo 1: Diagrama de clases del diseño



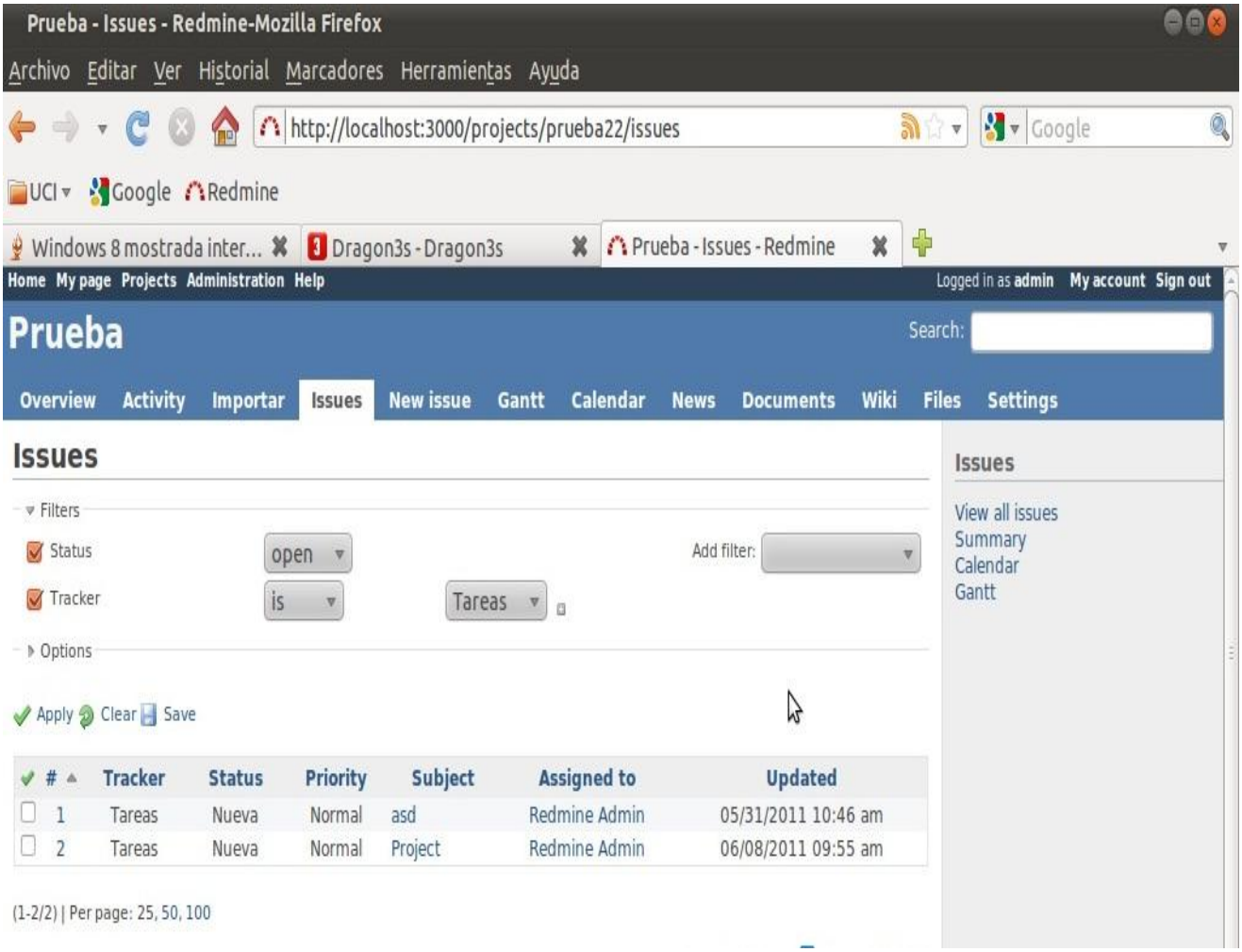
Anexo 2: Interfaz índex del plugin



Anexo 3: interfaz macht del plugin



Anexo 4: Interfaz result del plugin



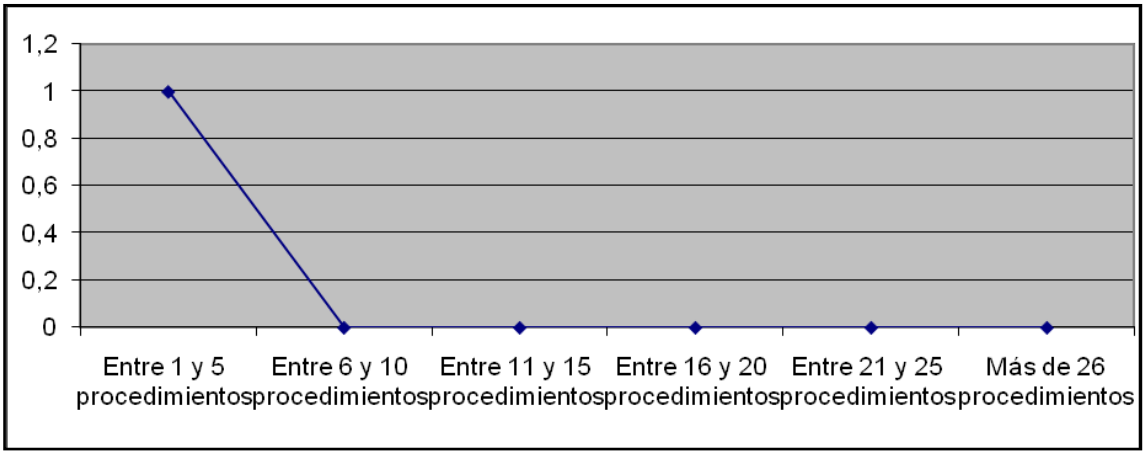
Anexo 5: Interfaz isseus del Redmine

Nombre: ImporterController	
Tipo de clase: Controladora	
Para cada responsabilidad	
Funcionalidad	**Descripción**
machtAction	Machear las tablas o campos de la petición entrada con los de la base de datos
resultAction	Insertar el elemento en base de datos y mostrar los resultados

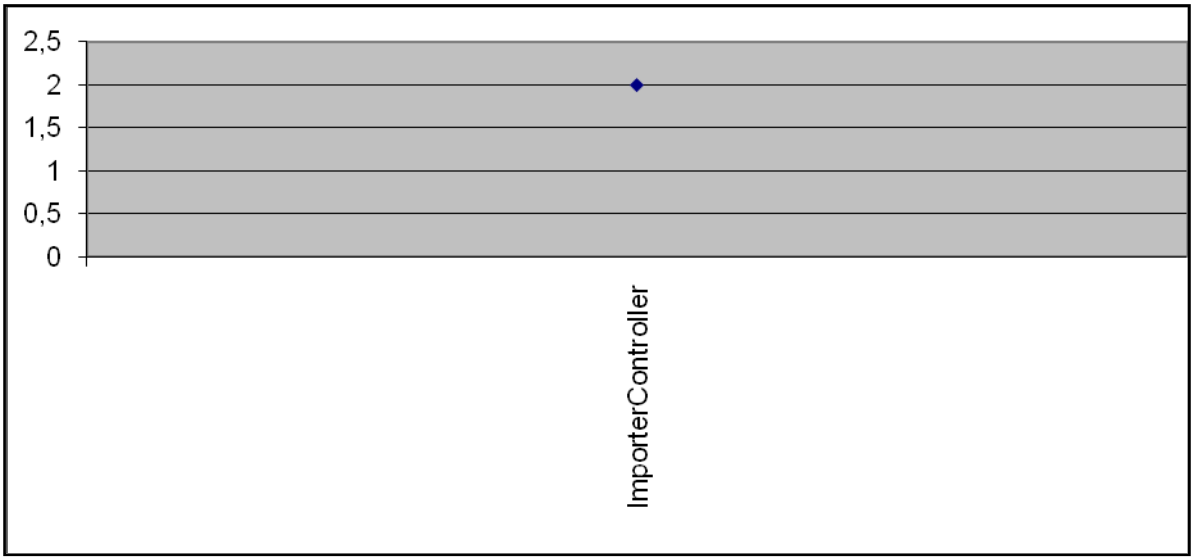
Anexo 6: Descripción de la clase ImporterController.

No	Subsistema	Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
1		ImporterController	2	Media	Media	Media
2		ImporterModel	1	Baja	Baja	Alta

Anexo 7: Instrumento de medición de la métrica Tamaño Operacional de Clase (TOC).



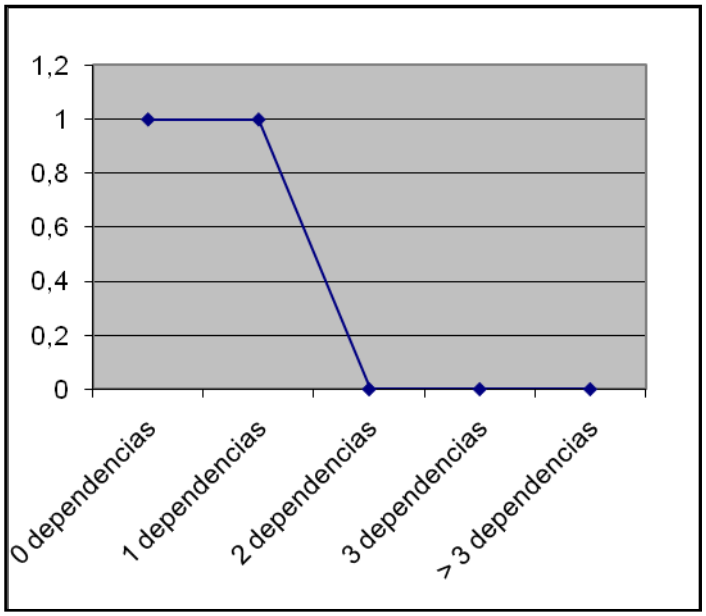
Anexo 8: Representación de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.



Anexo 9: Gráfica de los resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad (Complejidad de Implementación y Reutilización).

No	Subsistema	Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mant.	Reutilización	Cantidad de Pruebas
1		ImporterController	1	Bajo	Baja	Alta	Baja
2		ImporterModel	0	Ninguno	Baja	Alta	Baja

Anexo 10: Resultados de la evaluación de la métrica RC y su influencia en los atributos de calidad (Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas).



Anexo 11: Representación de los resultados de la evaluación de la métrica RC agrupados por la tendencia de los valores.

Glosario de términos

Actividad: define un conjunto de tareas, funciones y definiciones, agrupadas bajo un mismo contexto, teniendo en cuenta la relación entre estas funciones, el lugar y el personal que las realiza, basados además en los principios del control interno.

Algoritmo: es un conjunto finito de instrucciones o pasos que sirven para ejecutar una tarea o resolver un problema.

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

Fichero: es una manera particular de codificar información para almacenarla en un archivo informático