Universidad de las Ciencias Informáticas
Facultad 6

Título: “Integración de la herramienta de gestión de proyecto Redmine con la suite colaborativa Zimbra”

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

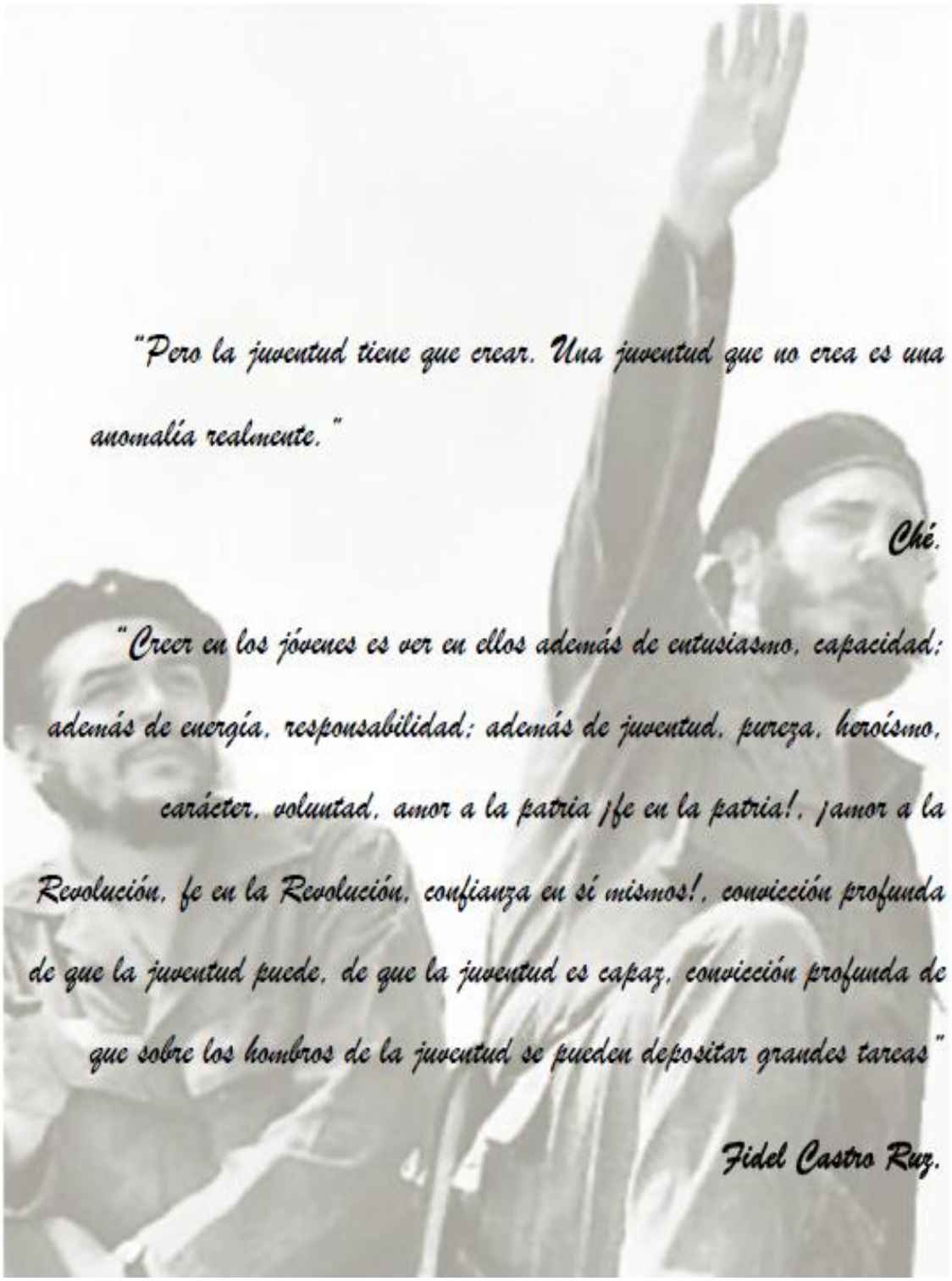
Autor: Yusnel Galiano Valdés

Tutor: MSc. Asnay Guirola González

Co-tutor: Ing. Javier Alfonso Valdés

La Habana, junio 2012.

“Año 54 de la Revolución”



"Pero la juventud tiene que crear. Una juventud que no crea es una anomalía realmente."

Ché.

"Creer en los jóvenes es ver en ellos además de entusiasmo, capacidad; además de energía, responsabilidad; además de juventud, pureza, heroísmo, carácter, voluntad, amor a la patria ¡fe en la patria!, ¡amor a la Revolución, fe en la Revolución, confianza en sí mismos!, convicción profunda de que la juventud puede, de que la juventud es capaz, convicción profunda de que sobre los hombros de la juventud se pueden depositar grandes tareas"

Fidel Castro Ruz.

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco 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 _____.

Yusnel Galiano Valdés

_____

Firma del Autor

MSc. Asnay Guirola González

_____

Firma del Tutor

Ing. Javier Alfonso Valdés

_____

Firma del Tutor

DATOS DE CONTACTO

TUTORES

Nombre: MSc. Asnay Guirola González
E-mail: aguirola@uci.cu
Años de graduados: 5 años de graduado.
Años de experiencia: 3 años de experiencia.
Categoría docente: profesor instructor.

Nombre: Ing. Javier Alfonso Valdés
E-mail: jalfonso@uci.cu
Años de graduados: 2 años de graduado.
Años de experiencia: 1 años de experiencia.
Categoría docente: profesor adiestrado.

AGRADECIMIENTOS

A Fidel Castro por crear esta escuela que me dio la oportunidad de hacer mis sueños realidad. A mis abuelos por todo lo que me inculcaron. A mis padres por darme la vida. A mí madre por siempre estar pendiente de mí cuando necesite algo. A mí familia, por cuidarme y sobre todo por confiar en mí. A mis amigos de la mini UCI de Artemisa por ser geniales y por siempre estar pendiente de cada paso que doy, a mí mejor amiga de los 5 años Geidy, que gracias a ella estoy aquí. A mis amigos de la UCI, especialmente por acompañarme y estar ahí cuando los necesitaba. A mi tutor Asnay que siempre estuvo ahí cuando tenía dudas, cuando el camino se tornaba difícil, por ser perfeccionista con cada detalle y hacerme dar lo mejor de mí en cada momento, a Nara Lidia por confiar en mí y ayudarme en todo el proceso de la tesis. A Yordanys que pese a que no es mi tutora me ayudo con la tesis. A los miembros del tribunal por cada sugerencia para que este trabajo llegara al final. A mis profesores desde primer año hasta quinto.

DEDICATORIA

A mis Padres Remberto y María Angélica a los cuales les debo todo lo que soy, por sus enseñanzas, su amor incondicional y por siempre haber confiado en mí. A mi madre, la cual amo con toda las fuerzas de mi corazón y le debo todo lo que hoy soy. A león por ser como un padre conmigo. A mi novia Georladys por quererme, apoyarme y estar ahí cuando más lo necesitaba. A mi hija Mía Fernanda por ser una niña buena y por inspirarme y darme fuerzas para seguir adelante. A mi mejor amigo el Duma por ser el mejor amigo de toda mi vida. A mi hermano, y a mis primos hermanos que también son como mis hermanos, a los más pequeños les dejo el ejemplo. A mis amigos por acompañarme en este viaje y hacerlo más ameno.

RESUMEN

La tarea de planificar y gestionar proyectos se ha ido desarrollando y haciéndose más compleja con el paso del tiempo, esto es debido al creciente flujo de información con el cual se interactúa de forma diaria, por lo que se han creado aplicaciones para la realización de este proceso. Este es el caso de la herramienta de gestión de proyecto Redmine, que realiza esta función a través de una interfaz web; sin embargo, esta aplicación no brinda todas las opciones que se pudieran necesitar como la integración con la herramientas colaborativa Zimbra.

Actualmente la tendencia para la resolución de los problemas del Redmine ha sido la creación de plugins a terceros o para el mismo sistema. El presente trabajo tiene como objetivo desarrollar un plugin que permita la integración entre Redmine y la suite colaborativa Zimbra en su versión 7.1.3, con el fin de sincronizar los calendarios de ambas. El plugin implementado se sometió a pruebas funcionales mediante las cuales se comprobó el correcto funcionamiento del mismo y el resultado alcanzado es de suma importancia pues beneficia directamente el proceso de desarrollo de software en el centro DATEC. Además, se realizó un período de pruebas para la validación del trabajo asegurándose que el sistema cumpla con las normas vigentes para el desarrollo.

Palabras claves:

Calendario, Integración, Plugin, Redmine, Sincronización, Zimbra.

.

ÍNDICE

Índice de Figuras

Índice de Tablas

INTRODUCCIÓN

A lo largo de la historia el hombre ha creado y desarrollado un conjunto de técnicas, procesos y máquinas para apoyar su capacidad de memoria, de pensamiento y comunicación. Las cuales han dado paso a la creación de la Informática. Esta ciencia estudia los métodos, procesos y técnicas desarrolladas en ordenadores con el objetivo de procesar, almacenar y trasmitir la información de forma digital.

Desde el inicio de la informática en el mundo se han creado un conjunto de aplicaciones para disímiles propósitos. Varias de estas tienen como objetivo la gestión de proyectos y en la actualidad son utilizadas en la gerencia, dirección o administración de proyectos. Dentro de las principales funcionalidades de la gestión de proyecto se encuentran, organizar y administrar recursos de manera tal que se pueda culminar todo el trabajo en el tiempo definido y con la calidad requerida.

En Cuba existen varios centros de producción de software, algunos de ellos se encuentran en la Universidad de las Ciencias Informáticas (UCI), con el fin de informatizar el país y desarrollar la industria del software. La universidad cuenta con una infraestructura dividida por varios departamentos, los cuales controlan todo el funcionamiento interno de cada proyecto. El desarrollo de la producción se concentra en varios centros de desarrollo y se destacan resultados en las esferas de salud, educación, software libre, sistemas legales, realidad virtual, automatización, bioinformática, procesamiento de imágenes, señales digitales y gestión de los datos.

En el entorno productivo utilizado en la UCI se emplea el Redmine como herramienta para la gestión de proyectos, dicho sistema es una aplicación web, de trabajo colaborativo, orientada a la coordinación de proyectos, principalmente de desarrollo de software, sirviendo como herramienta principal de comunicación entre los distintos componentes del proyecto. Esta es usada para mostrar y gestionar las tareas asignadas a cada estudiante; solo que para utilizar sus funcionalidades es necesario interactuar directamente con dicho sistema.

Otra herramienta que aprovecha la infraestructura de la universidad es la suite colaborativa de gestión de correo electrónico Zimbra. Esta aplicación cuenta con varias versiones disponibles: una versión soportada por la comunidad de software libre, y otra un poco mejorada con licencia propietaria. En ambos casos permite enviar, recibir, guardar y buscar los mensajes procesados cada día. Cuenta con herramientas auxiliares como: gestión de tareas y de agenda personal.

Introducción

La agenda de Zimbra permite controlar y programar citas, reuniones y eventos; y usa un formato estándar para el intercambio de datos de este tipo por Internet. Permite añadir tareas a la lista predeterminada de tareas y crear otras listas para organizar las tareas del usuario por actividades específicas, como por ejemplo, tareas profesionales y proyectos; además de sincronizar las tareas desde una lista de tareas remota en la nueva lista.

La dirección técnica de la UCI, así como los desarrolladores del centro DATEC han implementado un grupo de funcionalidades para personalizar y mejorar la usabilidad de Redmine, pese a esto aún quedan necesidades por cubrir, como agilizar y simplificar el proceso para consultar y gestionar las tareas asignadas en el Redmine.

Actualmente, cuando los usuarios desean darle seguimiento a las tareas que le han sido asignadas en sus áreas de trabajo, tienen que interactuar con el Redmine y llevar a cabo un conjunto de pasos engorrosos que requieren de tiempo y procesos, como filtrar las tareas asignadas. Además, los usuarios del Redmine no pueden revisar las tareas a través de Zimbra en el cual también se tienen tareas, eventos y citas. Todo esto provoca que un usuario tenga que interactuar con ambas herramientas, incluso cuando estas repliquen funcionalidades como las agendas de trabajo y la gestión de tareas.

Una vez descrita la **situación problemática** se define como **problema a resolver:** *¿Cómo simplificar el proceso de consulta de las tareas asignadas en la herramienta de gestión de proyectos Redmine?*

Por lo antes mencionado como **objeto de estudio** se define *el proceso de gestión de tareas que implementa la herramienta de gestión de proyecto Redmine* y como **campo de acción** *integración de la herramienta de gestión de proyectos Redmine con la suite colaborativa Zimbra para el seguimiento de las tareas.*

Se define como **objetivo general de la presente investigación**: *Desarrollar un plugin que permita la integración de los calendarios de la herramienta de gestión de proyectos Redmine con la suite colaborativa Zimbra.*

A partir de un análisis del objetivo general se derivan los siguientes objetivos específicos:

1. Realizar el análisis y diseño del plugin de integración entre Redmine y la suite de colaboración Zimbra.
2. Implementar el plugin de integración.

3. Validar el plugin de integración.

Con el fin de dar cumplimiento a los objetivos específicos se definieron como **tareas de la investigación**:

1. Realización de un análisis crítico y valorativo de las soluciones de integración de Redmine y Zimbra, existentes a nivel nacional e internacional, estableciendo similitudes con la investigación en curso.
2. Caracterización de la metodología, plataforma, tecnologías, y herramientas a utilizar en la investigación.
3. Caracterización de la arquitectura del Redmine, así como sus puntos de integración con sistemas legados.
4. Caracterización de los mecanismos de extensión de la herramienta Redmine.
5. Creación de los artefactos correspondientes a los Flujos de Trabajo: “Modelamiento del Negocio”, “Requerimientos”, “Análisis y Diseño” e “Implementación”.
6. Realización de los casos de prueba del plugin de integración.

Posibles resultados:

Un plugin de integración para la herramienta Redmine que facilita el seguimiento de las tareas y su estado actual desde la suite colaborativa Zimbra.

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, en el cual serán expuestos los principales conceptos asociados al problema en cuestión para lograr un mayor entendimiento por parte de los usuarios ajenos al sistema. Se muestran las principales características del Redmine así como las herramientas con las que se desea interoperar.

Capítulo 2 Análisis y diseño del sistema, en el cual se enfatiza la propuesta de solución para el problema de la investigación, así como la definición de conceptos vinculados a esta. Se muestra el modelo de dominio para explicar cómo será la interacción del sistema con el Redmine. Se enumeran los requisitos funcionales y no funcionales que debe cumplir la aplicación, además de la realización de una especificación de cada uno para una mejor interpretación de los mismos. Se describen los patrones utilizados, el diagrama de clases de diseño y de casos de uso para definir la estructura del plugin.

Capítulo 3 Implementación y prueba del sistema, se realiza todo lo relacionado con los flujos de trabajo de implementación y prueba, se muestra el diagrama de componentes para dar una visión de los principales componentes del sistema, artefactos y otros elementos así como las conexiones entre ellos y se realizan los casos de pruebas para garantizar la buena calidad del software.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se abordan conceptos relacionados con la herramienta de gestión de proyecto Redmine, con plugin y con la suite colaborativa Zimbra. Además, se describen la metodología, herramientas y tecnologías a utilizar.

1.1 Gestión de proyecto

La gestión de proyectos constituye uno de los elementos fundamentales en todo proceso de desarrollo del software. El éxito de un proyecto radica en un adecuado control y seguimiento del mismo. Cuando un proyecto se planifica se debe fiscalizar su desarrollo para comprobar que marcha sobre el plan previsto, es decir, debe satisfacer los objetivos de calidad, costo y tiempo. (1)

Se entiende la gestión de proyectos como la planificación, el seguimiento y el control de las actividades y de los recursos humanos y materiales que intervienen en el desarrollo de cualquier proyecto. (2)

Por lo que se concluye que la gestión de proyectos es el proceso que se encarga de organizar, administrar y dirigir los recursos para terminar en tiempo, con calidad y con un costo mínimo el trabajo requerido en el proyecto.

1.2 Redmine 1.4

Redmine es un gestor y planificador de proyectos con interfaz web, orientado a la coordinación de tareas, comunicación de participantes, y que puede especializarse en proyectos de desarrollo. (3)

Está desarrollado en Ruby on Rails. Soporta tanto MySQL como PostgreSQL o SQL Lite como gestores de bases de datos. La principal ventaja como gestor de proyectos es poder tener toda la información asociada a un proyecto acotada dentro del mismo. Permite el control de la ejecución del mismo, todo ello a través de su interfaz web. Por lo que Redmine se ha convertido en una de las herramientas de gestión de proyectos más usadas en el mundo actual, sus principales características son. (4)

Gestión de múltiples proyectos: Redmine permite gestionar múltiples proyectos desde una sola interfaz. La navegación es muy sencilla y se puede saltar y cambiar de proyecto en cualquier momento. Cada proyecto, además, puede tener una configuración totalmente diferente, el usuario puede tener un rol

distinto en cada uno y dentro del mismo pueden definirse varios subproyectos. Estos pueden definirse como privados, en los que el administrador debe dar acceso a cada miembro, o públicos siendo visibles para todo el mundo.

Personalización de proyectos: El Redmine permite que cada proyecto sea totalmente personalizable, lo que hace posible encontrar proyectos muy distintos entre sí, según sus objetivos. Lo más importante son los módulos que se pueden desactivar o activar para cada proyecto como: wiki, foro, noticias, peticiones, control del tiempo, documentos, ficheros o repositorio, aunque hay módulos comunes a todos los proyectos como el de actividad y vistazo. Si un proyecto está enfocado a notificar incidencias, se puede configurar para incluir solamente peticiones; si se busca un proyecto más colaborativo, la wiki y las noticias son una buena opción, e incluso se puede habilitar un proyecto solo con un foro.

Sistema flexible de seguimiento de tareas: Una de las funcionalidades más útiles para el desarrollo de un proyecto en Redmine son las peticiones y su visualización. Estas peticiones se dividen en tres tipos iniciales (errores, tareas y soporte) y pueden asignarse a un miembro del proyecto. Se puede indicar una fecha de inicio y fin para cada petición, e incluso llevar un control del tiempo y porcentaje realizado. También se le puede asignar una prioridad, enlazar con la subida de un fichero, y encajar en una categoría. Con todos estos datos, pueden visualizarse las peticiones de manera personalizada estableciendo filtros, y servir así de informes de tareas o incidencias. Además, dentro de un proyecto pueden establecerse versiones y asignar tareas a determinadas versiones, así conforme se marquen tareas completadas, las versiones irán completando su porcentaje automáticamente.

Integración en repositorios de código: Redmine puede integrarse con un repositorio de código (Subversion, Git, CVS, entre otros), la aplicación sirve así de interfaz web para el seguimiento del desarrollo de un proyecto. Pueden descargarse los ficheros, ver el historial, los cambios, e incluso descargar un archivo a modo de parche para aplicar a código desactualizado. Es un sistema de seguimiento de versiones, aunque no pueden actualizarse los ficheros directamente.

Uso de calendario y diagrama de Gantt: Redmine incluye un calendario para visualizar todas las peticiones a lo largo de un mes elegido, marcando claramente el día de inicio y 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 electrónico a todos los proyectos, definiendo antes los eventos que activan estos avisos. Además, cada usuario en su configuración puede elegir recibir notificaciones de cualquier evento, o solo las relacionadas con él (por ejemplo uno de los campos de las peticiones son las personas en seguimiento). Puede configurarse el servidor de correo entrante, permitiendo así actualizar peticiones simplemente por email e incluso crear nuevas peticiones. Todas las actividades de cada proyecto también pueden exportarse en Atom, para ser seguida desde un lector RSS. Si ninguna de estas opciones es favorita, en todos los proyectos existe el módulo de "Actividad", que refleja todo el flujo por días en el proyecto y lo muestra en una lista.

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, permitiendo imprimir posteriormente en un formato organizado. Las páginas de la wiki en cambio, pueden exportarse en HTML o TXT.

Otras características: Dentro de las funcionalidades que habría que destacar son la página personal de cada usuario, que ofrece una vista personalizable con información de todos los proyectos donde está participando, como un calendario global, o peticiones asignadas. Permite subir ficheros, definir campos personalizados para cada módulo, usar la barra de búsqueda global, ampliar la funcionalidad con decenas de extensiones y admite como bases de datos MySQL, SQLite y PostgreSQL.

Fallos y/o carencias importantes: Redmine es un gestor de proyectos muy potente y maduro, a pesar de su "poco" tiempo de desarrollo. Está inspirado en trac, por lo que estamos ante una herramienta muy flexible y bastante estable. Durante su uso, puede ser contradictorio sacarle carencias, siempre dependiendo de lo que esté buscando cada usuario, por ello hay que tener muy claro lo que se busca en este tipo de herramientas.

1.3 Plugin

Un plugin es una aplicación informática que añade funcionalidades específicas a un programa principal. Permite extender las capacidades de un programa de un modo determinado, por ejemplo la capacidad de mostrar videos, audio, ficheros de un determinado formato como ficheros PDF, presentaciones de ASAP, fichero VRML, entre otros. (5)

Estos programas son conocidos también como complementos, extensiones y addons (del inglés add-on, “agregado”). Su nombre procede del inglés (plug-in significa “enchufable”) y su presencia es muy habitual en los navegadores web, en reproductores de música y en sistemas de gestión de contenidos.

1.4 Plugins del Redmine

Redmine cuenta con una librería de múltiples plugins que añaden funcionalidades y una guía que da información de cómo desarrollarlos. Esta librería ha sido creada entre otros por los usuarios y la comunidad del Redmine para satisfacer algunas de sus necesidades. Cada plugin tiene una ficha específica donde se visualiza una pequeña descripción, versión disponible, como instalarlo y de donde obtener nuevas versiones.

Son muy variados y se adaptan a distintas necesidades. Dentro de ellos se destacan los dedicados a generar gráficos (Chart), a establecer tiempos para cada tarea (Timesheet) o a crear salas de chat (Chat). También son interesantes los plugins de terceros, extensiones que usan otras herramientas para integrarse con Redmine. (6)

1.5 Zimbra 7.1.3

Zimbra es un gestor de correo electrónico y calendario de código abierto, que ofrece un servicio de correo electrónico seguro y fiable. Incluye funciones como la agenda electrónica, maletín, libreta de direcciones, tareas y la escritura de documentos web.

Tiene como principales características: (7)

- Flexibilidad, permite personalizar sus funciones según las necesidades de los usuarios.
- Libertad, utiliza el cliente web de Zimbra junto con otros programas tradicionales, como plataforma mixta.
- Durabilidad, un servidor de correo electrónico y calendario extraordinariamente fiable y ampliable
- Bajo mantenimiento, gestión completamente sencilla.

- Cliente web basado en Ajax que incluye correo electrónico, contactos, calendario compartido, VoIP, aplicaciones y datos "mezclados" de diversas fuentes de Internet para empresas, y también de autoría de documentos web; todo incluido en el navegador web.
- Compatibilidad con aplicaciones de escritorio: sincronización propia entre ZCS y Microsoft Outlook, Entourage, Apple Mail, Libreta de direcciones e iCal; soporte completo de aplicaciones IMAP/POP.
- Zimbra para móviles: sincronización propia a través del aire con dispositivos de Windows Mobile, Symbian y Palm, sin necesidad de un servidor adicional.
- Servidores ZCS Linux y Mac OS X: con agente de transporte de correo (MTA), antispam, antivirus, directorio, base de datos, herramientas de migración, y consola de administración web basada en Ajax.

1.6 Metodologías de desarrollo de software.

Una metodología es un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software. Puede seguir uno o varios modelos de ciclo de vida. Las metodologías de desarrollo de software se dividen en dos grandes grupos, las pesadas o tradicionales y las ágiles. Las metodologías tradicionales se basan en la idea que el éxito del producto se puede lograr si se tiene todo correctamente documentado, mientras las ágiles defienden la idea de que el proceso de desarrollo del software, se centre en la solución final y no en la documentación alrededor de este, teniendo en cuenta solo la documentación necesaria y de forma muy sencilla. (8)

1.6.1 Open UP

Proceso Unificado Abierto (OpenUP) es una parte del Eclipse Process Framework (EPF), un proceso de código abierto desarrollado en el marco de Eclipse. Está desarrollado para pequeños equipos organizados. Es un proceso iterativo que es mínimo, completo, y extensible. Mantiene las características esenciales de la metodología RUP, como el desarrollo iterativo, casos de uso y escenarios de conducción de desarrollo, gestión de riesgos, y el enfoque centrado en la arquitectura. Está organizado dentro de cuatro áreas principales de contenido: Comunicación y Colaboración, Intención, Solución, y Administración. (9)

OpenUP/Basic es un proceso iterativo cuyas iteraciones se distribuyen a través de cuatro fases: Concepción, Elaboración, Construcción, y Transición. Cada fase podrá tener tantas iteraciones como se requiera, dependiendo del grado de novedad del dominio de negocio, de la tecnología a ser utilizada, de la complejidad de la arquitectura de la solución y del tamaño del proyecto, entre otros factores. (9)

Dentro de las principales características se encuentran: (9)

- Balance para confrontar las prioridades (necesidades y costos técnicos) para maximizar el valor para los stakeholders.
- Colaboración para alinear los intereses y un entendimiento compartido.
- Enfoque en articular la arquitectura para facilitar la colaboración técnica, reducir los riesgos y minimizar excesos y trabajo extra.
- Evolución continua para reducir riesgos, demostrar resultados y obtener retroalimentación de los clientes.

OpenUP puede dirigir cualquier tipo de proyecto, el tiempo de desarrollo es corto, el equipo de trabajo es pequeño y cuenta con menos de veinte artefactos, por estas razones es la herramienta idónea para el desarrollo de este trabajo.

1.7 Herramientas y Tecnologías

Para el desarrollo del plugin de integración es necesario tener en cuenta, las herramientas y tecnologías con las que se va a realizar el mismo.

1.7.1 Visual Paradigm 8.0

Es una herramienta CASE de modelado visual capaz de construir todos los tipos de diagramas de UML. Es compatible con una amplia gestión de casos de uso. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de buena calidad y menor coste. Permite construir todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. (10)

1.7.2 UML (Unified Modeling Languaje) 2.0

El Lenguaje de Modelamiento Unificado (UML Unified Modeling Language) es el lenguaje de modelado más conocido y utilizado en la actualidad; validado por la OMG (Object Management Group – Grupo de Gestión de Objeto). Es un lenguaje gráfico para visualizar, especificar y documentar las partes que comprende el desarrollo de un software. Permite modelar las clases de un lenguaje determinado, los esquemas de las bases de datos y los componentes de software. (11)

Dentro de sus principales características se encuentran: (11)

- Tecnología de orientación a objetos.
- Viabilidad en la corrección de errores.
- Desarrollo incremental e iterativo.

UML es fácil de aprender, permite una comunicación fluida entre los diversos actores del modelo, aunque suele estar más orientado a diseñadores de software y arquitectos del sistema.

1.7.3 NetBeans 7.1

NetBeans es un proyecto de código abierto que fue creado por Sun Micro Systems en junio del 2000. En la actualidad existen dos tipos de productos el NetBeansPlatform y el NetBeans como Entorno de desarrollo Integrado (IDE). Este último es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe un número importante de módulos para extender el NetBeans IDE. (12)

NetBeans Platform es una base modular y extensible usada como estructura de integración para crear grandes aplicaciones de escritorio. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones. (12)

Se opta por el uso de este IDE en su versión 7.1 porque es libre y de código abierto. Tiene una interfaz muy amigable e intuitiva, característica fundamental en un IDE. Aunque consume recursos adicionales con respecto a otros IDEs, es uno de los más utilizados en cuanto a desarrollo de aplicaciones web. Presenta un editor de formularios muy potente y muestra buena operatividad con Ruby on Rails.

1.7.4 Ruby on Rails 2.3

Un framework es un conjunto de bibliotecas, una plataforma, un entorno y un marco de trabajo el cual está orientado al desarrollo de aplicaciones que reutilizan a gran escala los componentes de software. Ruby on Rails es un framework de código abierto, para el desarrollo de aplicaciones web, creado por David Heine Meier Hansson. La primera versión fue liberada en el 2004 y lo implementó en una aplicación orientada a la administración de proyectos llamada Basecamp. Está basado en el modelo-vista-controlador (MVC).

Ruby on Rails, también conocido como (Rails o RoR) está basado en estos principios de desarrollo: (13)

- No lo vuelvas a repetir.
- Convención antes que configuración.

Primer principio:

No lo vuelvas a repetir (“Don’t Repeat Yourself”), es uno de los principios más novedosos que se puede encontrar en este framework. Esta regla permite tener un formulario y llamarlo las veces que se requiera y desde donde se necesite, simplemente con una línea de código. Permite manipular los registros como un objeto y a sus campos como un atributo en una tabla en la base de datos, sin necesidad de declarar nada.

Segundo principio:

Convención antes que Configuración (“Convention Over Configuration”), es un principio que plantea, que solo debe existir configuración para aquello que no escape de lo convencional. Para lograr esto el framework plantea un camino convencional con el cual se evita la configuración, cuando el comportamiento deseado escapa del convencional debe aplicar una configuración especial.

1.7.5 Ruby

Ruby es un lenguaje de código abierto, enfocado en la simplicidad y productividad, es orientado a objetos. Fue creado por Yukihiro “matz” Matsumoto, a menudo ha manifestado que “está tratando de hacer que Ruby sea natural, no simple”, de una forma que se asemeje a la vida real. “Ruby es simple en apariencia, pero complejo por dentro”. (14)

En Ruby todo es un objeto, se le puede asignar acciones y propiedades a toda información y código. Ruby trata de no restringir al desarrollador, permitiendo al usuario agregar o quitar funcionalidades a partes ya existentes, por lo que se considera un lenguaje flexible. Es fácilmente portable. Se desarrolla

mayoritariamente en GNU/Linux, pero corre en varios sistemas operativos como: UNIX, Mac OS X, Windows 95/98/Me/NT/2000/XP, BeOS, MS-DOS y OS/2.

Ruby tiene un conjunto de funcionalidades entre las que se encuentran las siguientes. (14)

- Manejo de excepciones, como Java y Python, para facilitar el manejo de errores.
- Un verdadero mark-and-sweep garbage collector para todos los objetos de Ruby. No es necesario mantener contadores de referencias en bibliotecas externas.
- Escribir extenciones en C para Ruby es más fácil que hacer lo mismo para Perl o Python, con una API muy elegante para utilizar Ruby desde C. Esto incluye llamadas para embeber Ruby en otros programas, y así usarlo como lenguaje de scripting. También está disponible la interfaz SWIG.
- Puede cargar bibliotecas de extensión dinámicamente si lo permite el sistema operativo.
- Tiene manejo de hilos (threading) independiente del sistema operativo. De esta forma, tiene soporte multi-hilo en todas las plataformas en las que corre Ruby, sin importar si el sistema operativo lo soporta o no, incluso en MS-DOS.

1.7.6 PostgreSQL 8.4

El Sistema Gestor de Bases de Datos Relacionales Orientadas a Objetos conocido como PostgreSQL está derivado del paquete Postgres escrito en Berkeley. Con cerca de una década de desarrollo tras él, PostgreSQL es el gestor de bases de datos de código abierto más avanzado hoy en día, ofreciendo control de concurrencia multi-versión, soportando casi toda la sintaxis SQL (incluyendo subconsultas, transacciones y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación (incluyendo C, C++, Java, perl, tcl, python y Ruby). Es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD. (15)

Presenta grandes ventajas y características como son: alta concurrencia y mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo commit. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases de datos, eliminando la necesidad del uso de bloqueos explícitos. Se han implementado importantes características del motor de datos,

incluyendo subconsultas, valores por defecto, restricciones a valores en los campos (constraints) y disparadores (triggers). (16)

Conclusiones Parciales

En la investigación realizada, no se encontró ningún plugin que cumpla con las características necesarias para dar respuesta al problema planteado, aunque si existen diversos plugin para otras o similares funcionalidades. También se abordaron los principales conceptos relacionados con la investigación. Se definieron las metodologías, tecnologías y herramientas con las cuales se desarrollará la solución, siendo estas, para el diseño OpenUp como metodología a emplear, UML 2.0 como lenguaje de modelado y Visual Paradigm 8.0 como herramienta case. Para la implementación se empleó Ruby on Rails 2.3 como framework de desarrollo empleando como lenguaje de programación Ruby e IDE el NetBeans 7.1. Para la gestión de los datos se empleó el gestor de base de datos PostgresSQL 8.4.

CAPÍTULO 2: ANÁLISIS Y DISEÑO.

Introducción

Este capítulo enfatiza en la propuesta de solución para el problema de la investigación, así como la definición de conceptos vinculados a esta. Se muestra el modelo de dominio para explicar cómo será la interacción del sistema con el Redmine. Se enumeran los requisitos funcionales y no funcionales que debe cumplir la aplicación, además de la realización de una especificación de cada uno para una mejor interpretación de los mismos. Se describen los patrones utilizados, el diagrama de clases de diseño y de casos de uso para definir la estructura del plugin.

2.1 Modelo de Dominio

Se llama modelo del dominio a la representación visual de los conceptos u objetos del mundo real en un dominio de interés. Este modelo agrupa los conceptos de un dominio. Es el mecanismo fundamental para comprender el dominio del problema y para establecer conceptos comunes. (17)

Para proporcionar un mejor entendimiento de los principales conceptos que se manejan en el negocio y se pueda realizar este proceso de comprensión de la manera más cómoda y entendible, se decidió realizar el modelo de dominio, teniendo en cuenta los objetos existentes que de una forma u otra están relacionados con el sistema a desarrollar, estableciendo las posibles relaciones que existen entre ellos. El modelo de dominio se describe mediante diagramas de UML, especialmente mediante diagramas de clases. En la siguiente figura 1 se representa el modelo de dominio realizado.

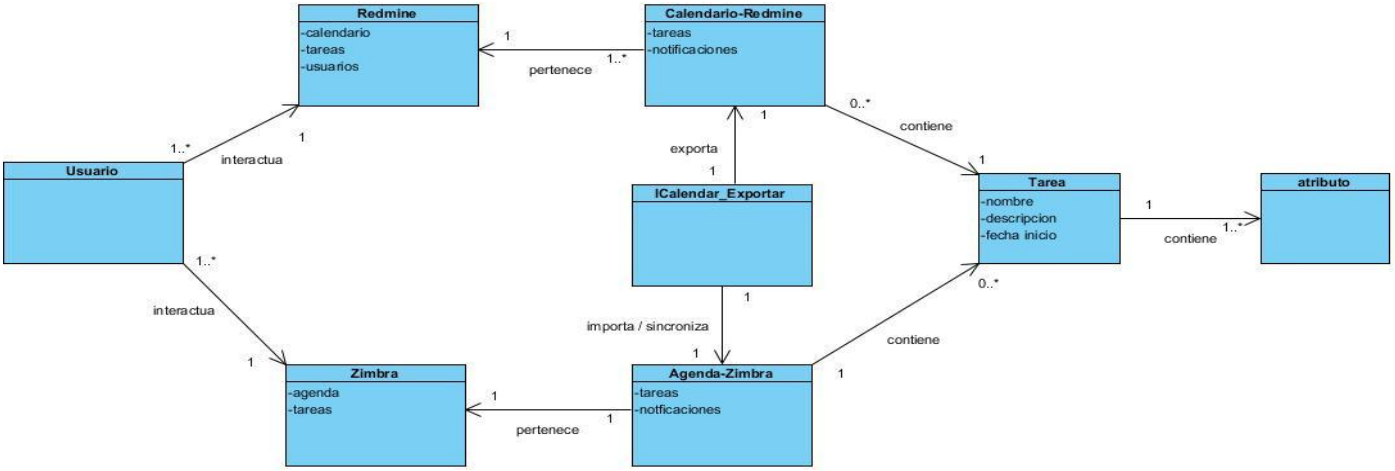


Figura 1. Modelo de Dominio

A continuación se presentan una descripción de los conceptos que son de importancia para el desarrollo del plugin.

Usuario:

Persona responsable y capacitada para utilizar el sistema de gestión de proyecto Redmine y el plugin para exportar el calendario.

Redmine:

Es una herramienta para la gestión de proyecto. Es una aplicación web, de trabajo colaborativo, orientada a la coordinación de proyectos, principalmente de desarrollo de software, sirviendo como herramienta principal de comunicación entre los distintos componentes del proyecto.

Zimbra:

Es un gestor de correo electrónico y calendario de código abierto, que ofrece un servicio de correo electrónico seguro y fiable.

Calendario – Redmine:

Tiene como objetivo principal la organización y estructuración del tiempo y su transcurrir. El calendario de Redmine es una forma visual que permite ver todas las peticiones asignadas a cada usuario con el tiempo de inicio y tiempo fin de cada tarea.

Agenda Zimbra:

La agenda de Zimbra es un calendario que permite visualizar, controlar y programar las citas, reuniones y eventos.

Tareas

Son las tareas asignadas a cada usuario en el Redmine que van a ser exportadas en el calendario de Redmine hacia la agenda de Zimbra.

Atributos

Son los atributos específicos como fecha inicio, fecha fin, descripción y resumen de cada tarea asignada a los usuarios en el Redmine.

2.2 Propuesta de Solución

Como propuesta de solución se presenta la creación de un plugin para integrar la herramienta de gestión de proyecto Redmine con la suite de colaborativa de correo Zimbra. Dicho plugin debe exportar el calendario de la herramienta de gestión de proyecto Redmine en el formato estándar iCalendar con extensión de archivo para calendarios "ICS". Se exporta en este formato ya que es el estándar utilizado por varios programas de correo electrónico como Zimbra, para el intercambio de información de calendario. El formato iCalendar conocido como “iCal” permite a los usuarios publicar y compartir información de calendario en la Web y por correo electrónico, a menudo se utiliza para enviar las solicitudes de reunión a otros usuarios, que pueden importar los eventos en sus propios calendarios. Estos archivos se guardan en un formato de texto plano, contienen información como el título, resumen, hora de inicio y hora de finalización del evento. De esta forma se crea un nuevo calendario en la agenda de Zimbra, se selecciona calendario remoto y se importa el calendario exportado del Redmine.

2.3 Requisitos

El proceso de construcción de un software está dado por disímiles actividades que han de ser monitoreadas por alguna guía metodológica, garantizando así la calidad del producto y la satisfacción de los clientes y desarrolladores al mismo tiempo. Durante la construcción del software se llevan a cabo un grupo de técnicas y procedimientos de acuerdo con la metodología seleccionada para la construcción de la solución, arrojando así la documentación relacionada con cada uno de los flujos de trabajo.

2.3.1 Requisitos funcionales

Los requisitos funcionales muestran y definen las funcionalidades que el software debe realizar. Son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requisitos muestran las necesidades del cliente en el sistema para resolver los problemas existentes. Con la representación de los conceptos más importantes en el Modelo de Dominio, se determinaron los requisitos funcionales y los no funcionales que debe cumplir el plugin.

Requisitos Funcionales:

RF 1. Exportar calendario.

RF 2. Enviar notificación.

RF 3. Importar calendario.

RF 4. Determinar tareas a exportar.

RF 5. Configurar el tiempo de notificación.

RF 6. Sincronizar calendario.

2.3.2 Requisitos no funcionales

Son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Los requisitos no funcionales forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto.

RNF 1 Usabilidad

El sistema podrá ser usado por personas con conocimientos básicos en el manejo de computadoras. Cuenta con una interfaz amigable e intuitiva que sirve de ayuda al usuario para una mejor interacción con las funcionalidades que brinda.

RNF 2 Fiabilidad

Para que el usuario pueda realizar las funcionalidades en el sistema debe estar autenticado y de acuerdo con el rol que le sea asignado tendrá los permisos asociados. Un usuario no autenticado solo podrá acceder las funcionalidades públicas.

RNF 3 Requisitos del Software

La construcción de la aplicación funcionará bajo los conceptos de arquitectura cliente/servidor. Por tanto, las estaciones de trabajo del usuario final y servidores deben tener como requisitos mínimos de software:

Para los servidores:

Sistemas operativos GNU/Linux o Windows Server 2000 o superior. Servidor Web Apache 2.0 o superior. MySQL, SQLite o PostgreSQL como Sistema Gestor de Base de Datos.

Para las PCs clientes:

Un Navegador como Mozilla Firefox, Chrome, Zafari u otro navegador que cumpla con los estándares W3C. Sistema operativo: GNU/Linux o Windows.

RNF 4 Requisitos del Hardware

Para las PCs clientes:

Al menos 256 MB de memoria RAM. Procesador 512 MHz como mínimo.

Para los servidores:

Se requiere tarjeta de red. Al menos 1 GB de RAM y 500 MB de disco duro. Procesador 3 GHz como mínimo.

RNF 5 Legales

El sistema debe ajustarse y regirse por la ley, decretos leyes, decretos, resoluciones y manuales (órdenes) establecidos, que norman los procesos que serán automatizados. La mayoría de las herramientas de desarrollo son libres y del resto, las licencias están avaladas. Como producto, se distribuye amparado bajo las normativas legales establecidas en el registro comercial emitido por las entidades jurídicas de la Universidad de las Ciencias Informáticas.

RNF 6 Seguridad

Los usuarios deben autenticarse antes de entrar al sistema. Garantizar el acceso controlado a la información, se mostrarán las interfaces a los usuarios, en dependencia de su nivel de acceso.

2.3.3 Actores del sistema

El actor del sistema es un usuario fuera del sistema que interactúa con él. A continuación se muestra en la siguiente tabla su justificación.

Nombre del Actor	Descripción
Usuario	Representa al usuario que va a hacer uso del sistema, y quien tiene la posibilidad de interactuar con todas las funcionalidades de éste.
Administrador	Es el encargado de definir las configuraciones generales del sistema.

Tabla 1. Actores del sistema

2.4 Diagrama de casos de uso del sistema

El diagrama de casos de uso muestra el modo en cómo un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan (operaciones o casos de uso). A partir de los requisitos funcionales identificados en la aplicación se diseñó el diagrama de casos de usos donde se muestran las relaciones entre los casos de usos y los actores del sistema. (18)

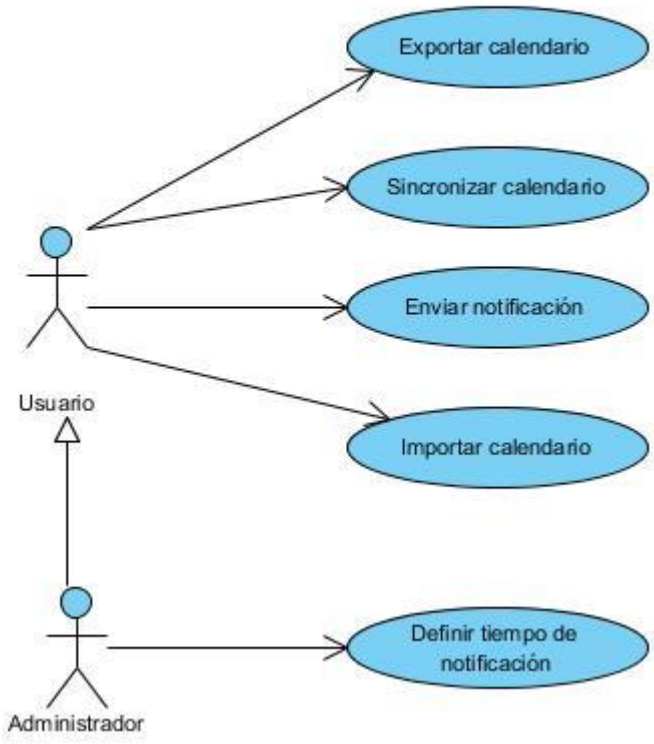


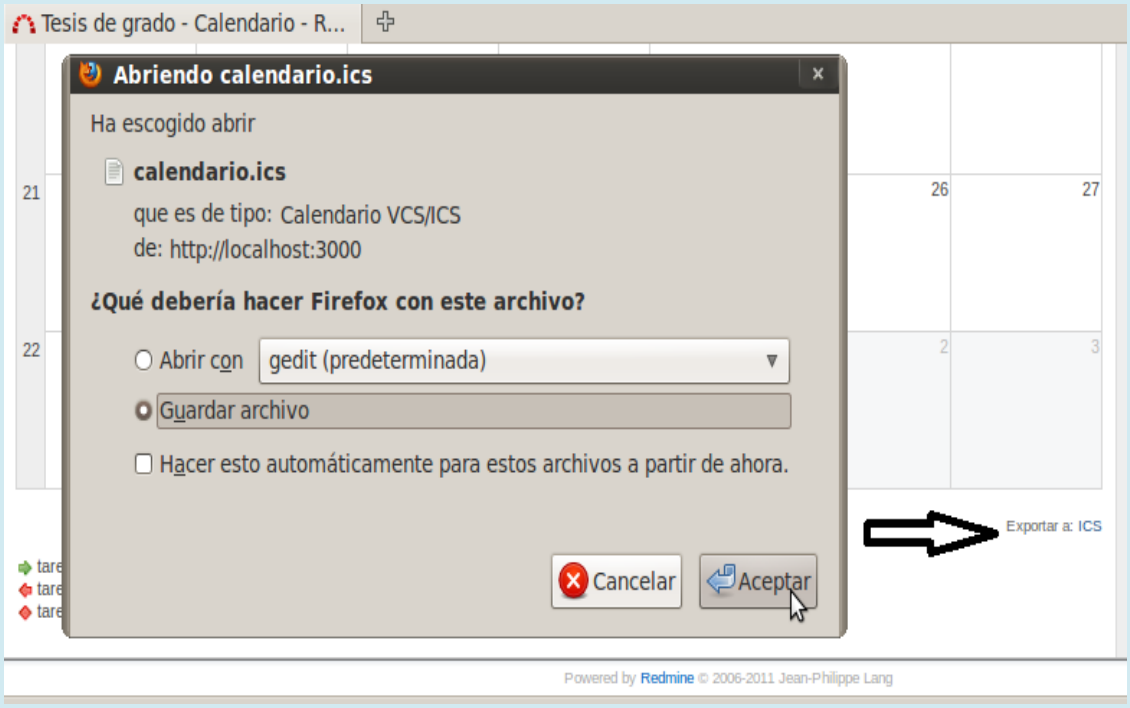
Figura 2. Diagrama de Casos de uso

2.4.1 Descripción de los casos de usos más significativos

Caso de uso 1: Exportar calendario

Objetivo	Exportar el calendario de Redmine en formato ICS.
Actores	Usuario
Resumen	El caso de uso inicia cuando el usuario selecciona la opción exportar a ICS o exportar detallado, este caso de uso le permite al usuario crear un calendario con todas sus tareas o con las tareas seleccionadas y almacenar su información en un fichero de tipo ICS, y termina cuando se ejecuta cualquiera de las dos operaciones.
Complejidad	Alta

Prioridad	Crítico	
Precondiciones	Estar habilitada la opción de mostrar calendario.	
Poscondiciones	Se exporta el calendario en formato ICS.	
Referencias	RF1, RF4	
Flujo Normal de Eventos		
Actor		**Sistema**
1	El caso de uso se inicia cuando el usuario selecciona la opción exportar a ICS o exportar detallado.	
2		El sistema realiza la operación según la opción seleccionada por el usuario. • Si selecciona exportar a ICS, ver sección “Exporta a ICS”. • Si selecciona exportar detallado ver sección “Exportar detallado”.
3		El caso de uso termina cuando el sistema exporta el calendario.
Sección 1: “Exportar a ICS”		
1	El usuario selecciona la opción “Exportar a ICS”.	
2		El sistema muestra una ventana que permite visualizar o guardar en una dirección el fichero del calendario.(Ver interfaz 1)
3	El usuario una vez completado todos los valores correspondientes a la creación del calendario da clic en el botón “Guardar”.	
4		El caso de uso termina cuando el sistema guarda todos los datos correspondientes al calendario en formato ICS.
Prototipo de Interfaz		



Interfaz 1

Sección 2: "Exportar detallado"		
1	El usuario selecciona la opción "Exportar detallado".	
2		El sistema muestra una interfaz con una tabla que visualiza todas las tareas del calendario como se muestra en la Interfaz 2.
3	El usuario selecciona las tareas a través del checkbox ID y da clic en el botón "Guardar".	
4		El sistema guarda las tareas seleccionadas para crear el nuevo calendario.
5	El usuario selecciona la opción "Exportar calendario"	

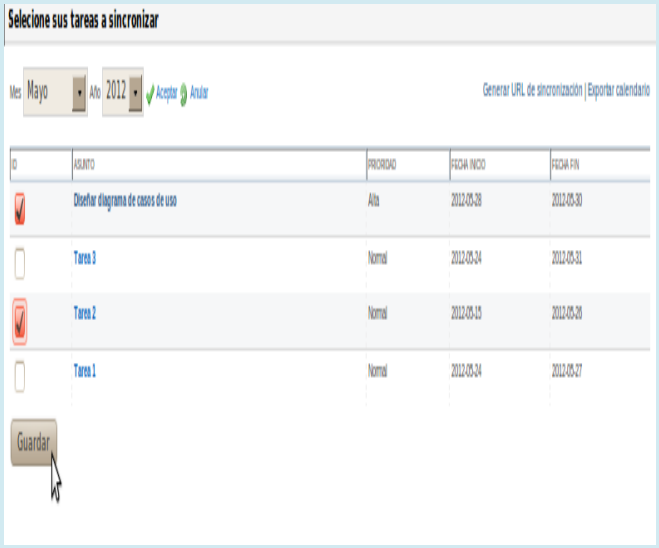
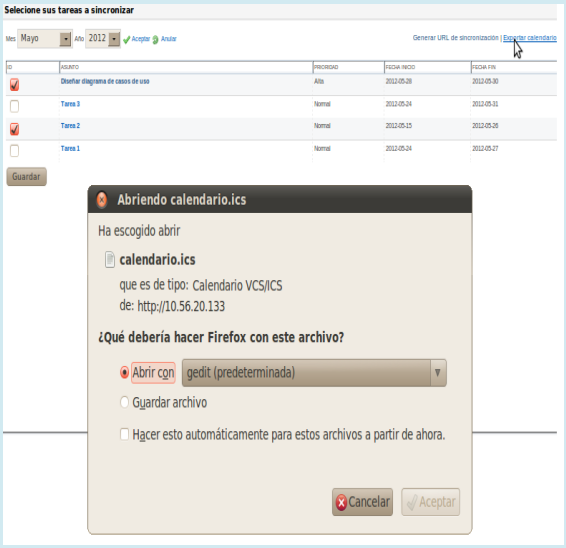
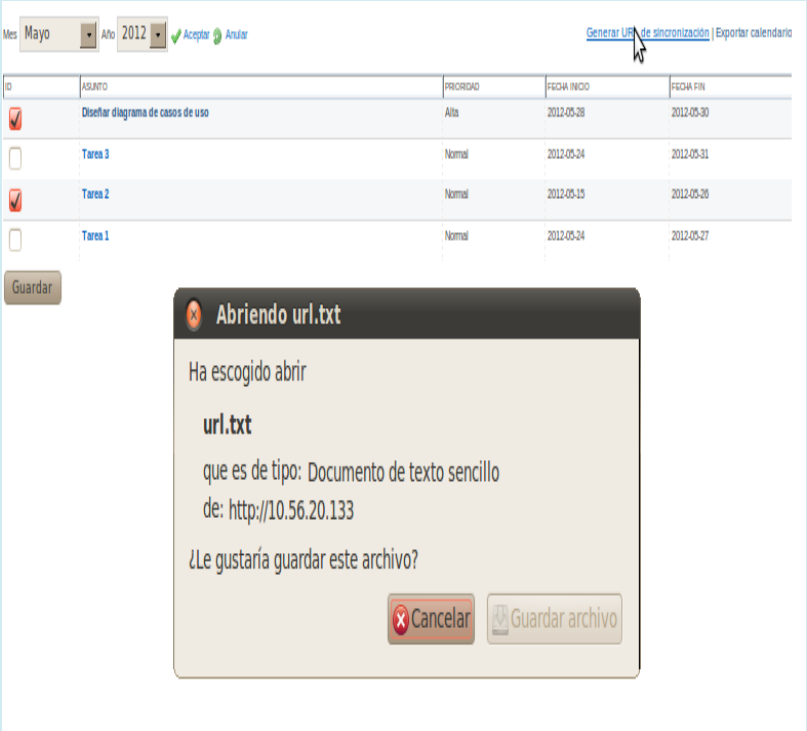
6		El sistema muestra una ventana que permite visualizar o guardar en una dirección el fichero del calendario. (Ver interfaz 3)
7	El usuario una vez completado todos los valores correspondientes a la creación del calendario da clic en el botón “Guardar”.	
8		El caso de uso termina cuando el sistema guarda todos los datos correspondientes al calendario en formato ICS.
Prototipo de Interfaz		
 Interfaz 2	 Interfaz 3	

Tabla 2. Caso de uso 1

Caso de uso 2: Sincronizar calendario

Objetivo	Sincronizar calendario

Actores	Usuario	
Resumen	El caso de uso se inicia cuando el usuario selecciona la opción generar URL, se exporta un fichero con la dirección URL del calendario y termina cuando el sistema exporta el fichero a la dirección indicada.	
Complejidad	Alta	
Prioridad	Crítico	
Precondiciones	El usuario debe seleccionar las tareas a sincronizar.	
Poscondiciones	Se exporta el fichero con la dirección URL a la dirección especificada.	
Referencias	RF4, RF6	
Flujo normal de eventos		
	Actor	**Sistema**
1	El usuario selecciona la opción “Generar URL”	
6		El sistema muestra una ventana que permite visualizar o guardar en una dirección el fichero que contiene la dirección URL del calendario. (Ver interfaz 4)
7	Selecciona “Guardar”	
8		El caso de uso termina cuando el sistema guarda la dirección URL del calendario en un fichero de tipo txt.
Prototipo de Interfaz		



interfaz 4

Tabla 3.Caso de uso 2

2.5 Diagrama de Clases del diseño

El diagrama de clases del diseño se encarga de representar los métodos y atributos de cada una de las clases del sistema para mostrar de forma simple la colaboración y las tareas de cada una de ellas en relación al sistema que conforman. Partiendo de la descripción detallada de los casos de uso del sistema, se modelaron los diagramas de clases del diseño.

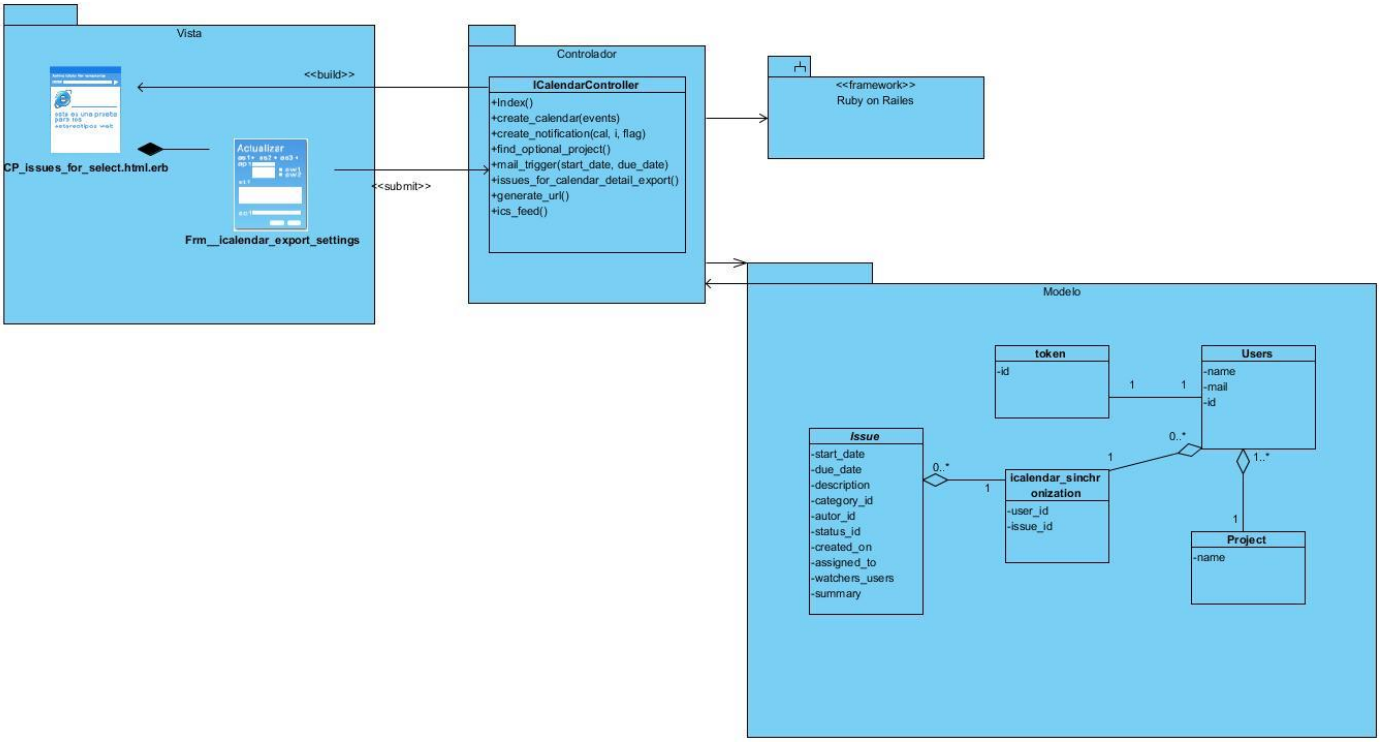


Figura 3.Diagrama de Clases del Diseño - CU Sincronizar Calendario

Nombre: ICalendar_Controller.rb	
Tipo de Clase: Controladora.	
Para cada responsabilidad:	
Nombre	**Descripción**
+ index()	Permite crear un calendario nuevo, hacer una consulta a la base de datos para obtener los atributos de los eventos que contiene el calendario. Exportar el calendario.

+ create_calendar(events)	Ofrece la posibilidad de asignarle al calendario nuevo los atributos obtenidos de la consulta que se le realiza a los eventos.
+ create_notification(cal,i,flag)	Permite crear una notificación y las alarmas que se van a enviar al correo.
+ find_optional_project ()	Define el proyecto en que el usuario está trabajando.
+ mail_trigger(start_date,due_date)	Define el tiempo con el cual llegará la notificación al correo.
+ issues_for_select()	Muestra las tareas del usuario en el proyecto en que se encuentra en ese momento. Esta lista de tarea será la que se mostrará para que el usuario defina cuáles de ellas serán sincronizadas.
+ issues_for_calendar_detail_export()	Crea el calendario detallo a partir de las tareas que fueron seleccionadas previamente.
+ ics_feed()	Método que crea y actualiza el calendario del usuario a partir de la conexión por URL desde el Zimbra.
+ generate_url()	Genera la URL por la cual se va a actualizar el calendario del usuario

	desde el Zimbra

Tabla 4. Descripción de los métodos de la clase controladora

2.6 Patrones arquitectónicos

Los patrones son de gran importancia para detallar las mejores prácticas, buenos diseños, y encapsulan la experiencia permitiendo reutilizar dicha experiencia. Constituyen mecanismos cuyo objetivo es la solución de problemas que ocurren repetidamente dentro de un contexto muy bien definido. MVC son las siglas de Modelo-Vista-Controlador. Es el patrón de diseño de software en el cual se basa la estructura de Redmine. El MVC considera dividir una aplicación en 3 partes bien diferenciadas, con funcionalidades bien definidas. Incluye varias características que permiten desarrollar de forma rápida y fácil las aplicaciones sofisticadas que utilizan los últimos estándares web. (19)

Modelo

Son los datos y las reglas del negocio. El modelo es un conjunto de clases que representan la información del mundo real, que puede tener varias vistas cada una con su correspondiente controlador. Este es responsable de: (20)

- Acceder a la capa de almacenamiento de datos.
- Definir las reglas del negocio (funcionalidad del sistema).
- Llevar un registro de las vistas y controladores del sistema.
- Notificar a las vistas los cambios que en los datos pueda producir un agente externo.

Controlador

El controlador manipula el modelo y gestiona las vistas. En una aplicación web es el código que obtiene datos dinámicamente y genera el código HTML.

Este es responsable de: (20)

- Recibir los eventos de entrada (Un ejemplo, un clic o un cambio en un campo de texto).
- Contener reglas de gestión de eventos, del tipo “Si evento Z, entonces acción W”.
- Permite realizar peticiones al modelo o a las vistas.

Vistas

Las vistas gestionan la presentación de la información de la aplicación. Esta se encarga de: *(21)*

- Recibir datos del modelo y la muestra al usuario.
- Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
- Pueden dar el servicio de actualización para que sea invocado por el controlador o por el modelo.

Se decidió seleccionar este tipo de estilo arquitectónico para organizar los principales componentes del sistema a desarrollar y representar las relaciones que existen entre ellos, ya que el plugin a desarrollar constituye una nueva funcionalidad que será añadida a la plataforma Redmine utilizando el framework de desarrollo Ruby on Rails el cual hace uso del patrón MVC.

Como patrón arquitectónico para representar el modelo se utilizó Active Record como mapeo relacional de objetos (ORM). Active Record, es un objeto que contiene una estructura de datos del registro en un recurso externo, como una fila en una tabla de base de datos y añade un poco de lógica de dominio para ese objeto, donde un objeto contiene los datos y el comportamiento del mismo. Gran parte de estos datos es persistente, y deben ser almacenados en una base de datos. Active Record utiliza el enfoque más obvio: poner la lógica de acceso a datos en el objeto del dominio. De esta manera todas las personas pueden leer y escribir los datos de la base de datos. (21)

2.7 Patrones de diseño GRASP.

En los patrones generales de software para asignación de responsabilidades (GRASP) se codifican algunos de los principios, que se aplican al diseñar los diagramas de interacción. Una de las principales ventajas a señalar en el uso del framework Ruby on Rails es que está basado en patrones de diseño. Los patrones de diseño empleados en la solución pertenecen al conjunto de patrones GRASP y GOF.

A continuación se explican los patrones utilizados: (21)

Experto: Consiste en asignar una responsabilidad al experto en información: que es la clase que cuenta con la información necesaria para cumplir dicha responsabilidad. Un ejemplo de ello es la clase controladora ICalendarController la cual garantiza que, si se desea crear un calendario, cuenta con la funcionalidad que permite realizar esta operación auxiliándose de la información obtenida de las clases modelo con las cuales se encuentra relacionada.

Se presenta un ejemplo de la clase ICalendarController donde se evidencia el patrón experto:

ICalendarController

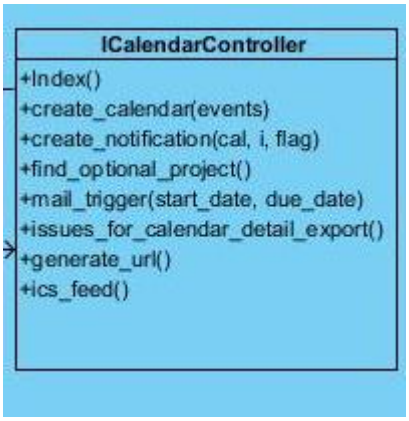


Figura 4. Clase controladora ICalendar_Controller

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 clases modelos y estas a su vez las entidades. Garantizando con ello la distribución por capas de la arquitectura. Se evidencia en la clase controladora ICalendarController en el método create_calendar(events), cuando se crea el objeto de tipo calendario.

Se presenta un ejemplo del método donde se evidencia el patrón creador:

create_calendar(events)

```
def create_calendar(events)
  cal = Icalendar::Calendar.new
  events.each { |i|
    if (i.due_date == nil) && (i.is_a? Issue) && (i.fixed_version != nil)
      i.due_date = i.fixed_version.due_date
    end
    if i.due_date == nil
      next
    end
    start_event = create_notification(cal,i,true)
    end_event = create_notification(cal,i,false)
    cal.add_event(start_event)
    cal.add_event(end_event)
  }
  return cal
end
```

Figura 5. Ejemplo del método create_calendar

Controlador: Las clases controladoras que son las encargadas de gestionar el acceso a la 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. El ejemplo de esta clase es ICalendarController que es la que contiene las principales funcionalidades del negocio.

Se presenta un ejemplo de una clase controladora donde se evidencia el patrón controlador:

ICalendarController

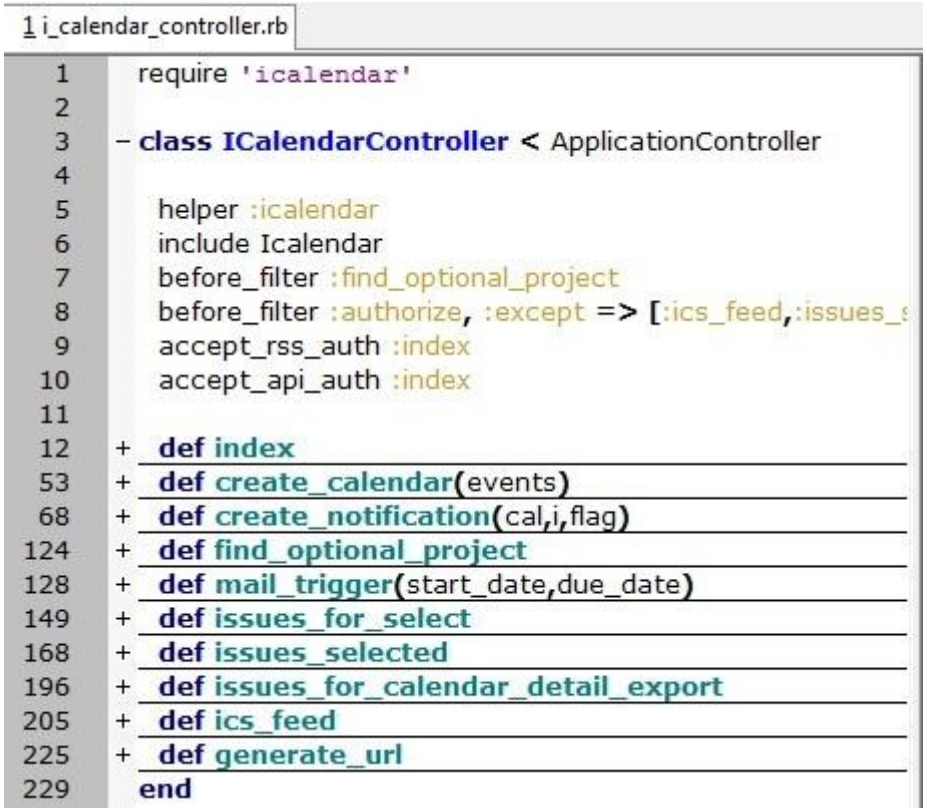


Figura 6. Ejemplo de la clase ICalendarController

Bajo acoplamiento: Las clases se diseñaron 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. Cuando las relaciones entre clases sean las mínimas indispensables y cada clase tiene solamente las funcionalidades necesarias a realizar.

A continuación se muestran las entidades del modelo donde se evidencian los patrones, Bajo Acoplamiento y Alta Cohesión:

Entidades del modelo

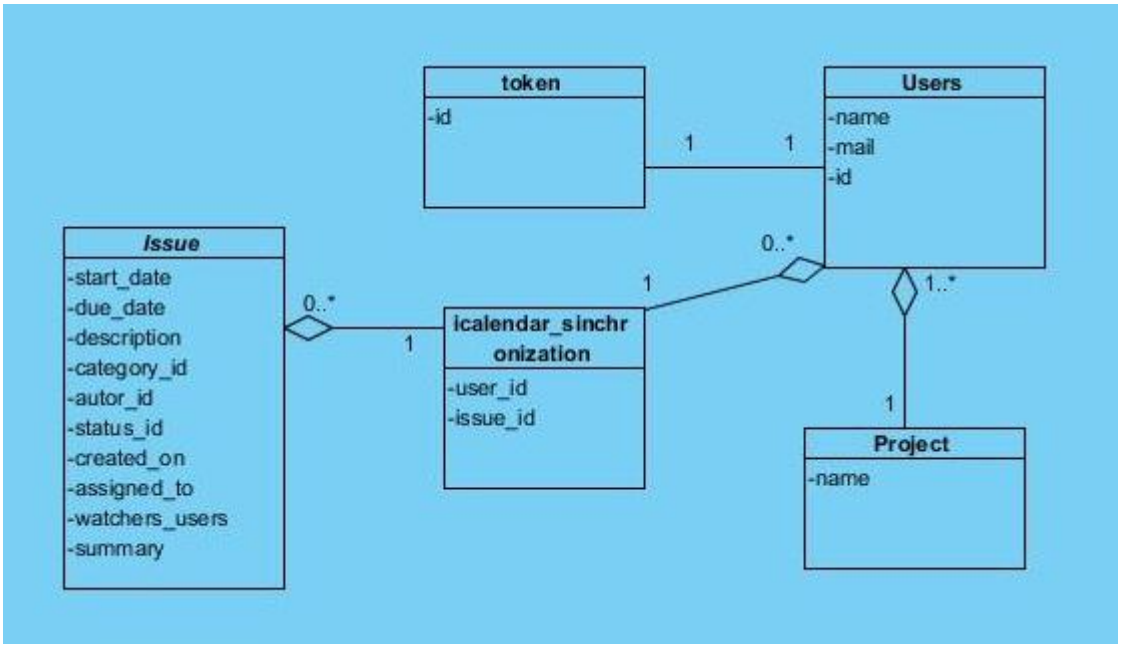


Figura 7. Entidades del modelo

Patrones de diseño GOF.

Los patrones de diseño GOF, son clasificados según su propósito en creacionales, estructurales y de composición, mientras que respecto a su ámbito se clasifican en clases y objetos. (20)

Según su propósito:

1. De Creación: Resuelven problemas relativos a la creación de objetos.
2. Estructurales: Resuelven problemas relativos a la composición de objetos.
3. De Comportamiento: Resuelven problemas relativos a la interacción entre objetos.

Decorador: Añade funcionalidad a una clase dinámicamente. Esto permite no tener que crear sucesivas clases que hereden de la primera incorporando la nueva funcionalidad, sino otras que la implementan y se asocian a la primera. Se evidencia en el uso del ORM Active Record en los métodos de acceso a datos.

Se presenta un ejemplo del método find_optional_project donde se evidencia el patrón decorador:

find_optional_project

```
def find_optional_project
  @project = Project.find_by_identifier(params[:project_id])
end
```

Figura 8. Método find_optional_project

Observador: Define una dependencia de uno a muchos entre objetos, de forma que cuando un objeto cambie de estado se notifiquen y actualicen automáticamente todos los objetos que dependen de él. Este es evidenciado dentro del sistema mediante los usos de los hooks (gancho).

Se presenta un ejemplo de un hooks que se define en view_calendar_show_bottom.html.erb la cual agrega información a la vista show.html.erb cuando esta es mostrada. Este efecto es posible porque la vista show.html.erb del calendario hace el lanzamiento de un hook durante su ejecución y view_calendar_show_bottom.html.erb tiene un hook en escucha esperando ser ejecutado. Por lo que siguiendo este patrón es muy sencillo hacer modificaciones y agregar información a las vistas del sistema sin tener necesidad de editar las plantillas bases de la aplicación.

view_calendars_show_bottom.html.erb

```
<% if User.current.allowed_to?(:view_calendar, @project, :global
=> true)%>
     <% other_formats_links do |f| %>
          <%= f.link_to 'ICS', :url => { :status => 'all',
:assigned_to => 'me', :controller => 'i_calendar', :action =>
'index', :project_id => @project, :key => User.current.rss_key,
:format => 'atom' } %>
     <% end %>
<% end %>
<%= link_to l(:view_calendar_advanced), :controller =>
'i_calendar', :action => 'issues_for_select',  :project_id =>
@project%>
```

Figura 9. Hook view_calendars_show_bottom

show.html.erb

```
<%= label_tag('year', l(:label_year)) %>
<%= select_year(@year, :prefix => "year", :discard_type =>
true) %>

<%= link_to_function l(:button_apply), '$("query_form").submit
()', :class => 'icon icon-checked' %>
<%= link_to l(:button_clear), { :project_id => @project,
:set_filter => 1 }, :class => 'icon icon-reload' %>
</p>
<% end %>

<%= error_messages_for 'query' %>
<% if @query.valid? %>
<%= render :partial => 'common/calendar', :locals =>
{:calendar => @calendar} %>
<%= call_hook(:view_calendars_show_bottom, { :year => @year,
:month => @month, :project => @project, :query => @query }) %>
```

Figura 10. Plantilla show.html.erb

2.8 Diagrama de secuencia

Los diagramas de secuencia ofrecen las interacciones entre objetos, ordenadas en secuencia temporal durante un escenario concreto. Describen el curso particular de los eventos de un caso de uso. A continuación se muestran los diagramas de secuencias de los casos de uso más significativos Exportar Calendario y Sincronizar Calendario.

En el presente diagrama se muestran el flujo de acciones que ocurren en la aplicación al ejecutar la acción “Exportar Calendario”.

Diagrama de Secuencia – Escenario: Exportar Calendario.

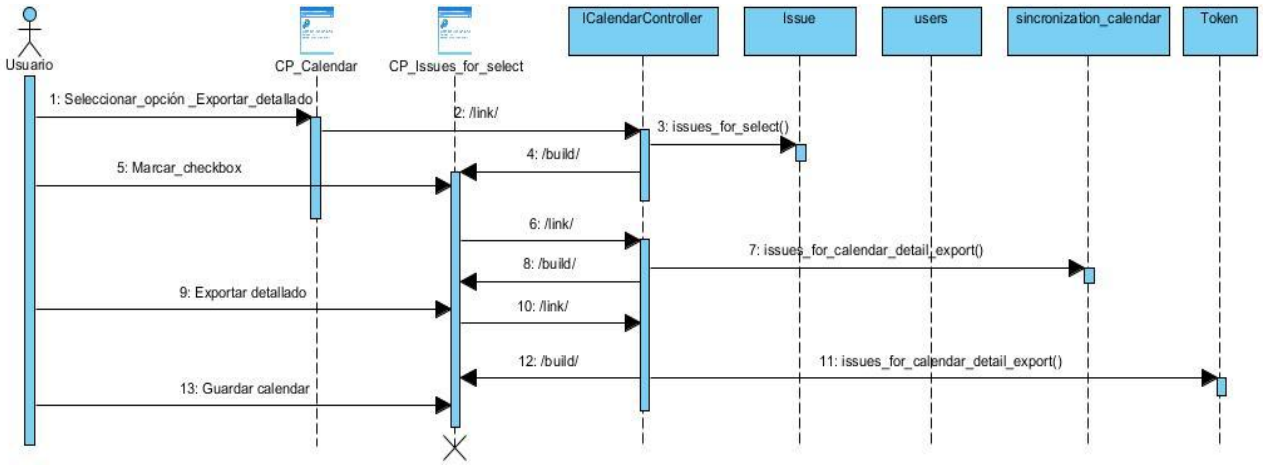


Figura 11. Diagrama de secuencia – Escenario: Exportar Calendario

En el presente diagrama se muestran el flujo de acciones que ocurren en la aplicación al ejecutar la acción "Sincronizar Calendario".

Diagrama de Secuencia – Escenario: Sincronizar Calendario.

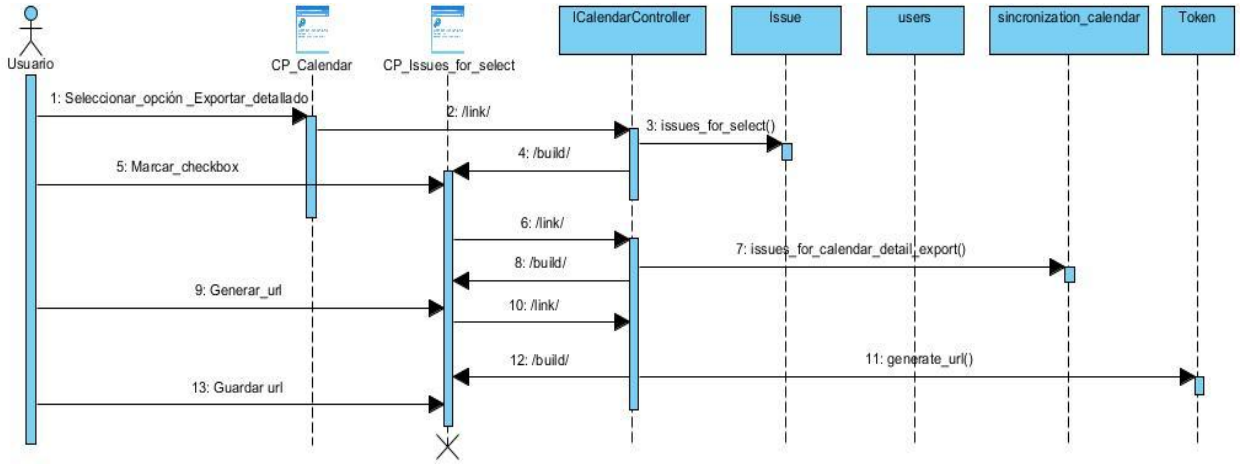


Figura 12. Diagrama de Secuencia – Escenario: Sincronizar Calendario

Conclusiones parciales

En el presente capítulo se definieron los conceptos más importantes para el desarrollo de la aplicación así como la especificación de los requisitos funcionales y no funcionales para el correcto funcionamiento del

sistema, identificándose 6 requisitos funcionales agrupados en 5 casos de usos, relacionados con 2 actores del sistema, el usuario y el administrador conformando el diagrama de caso de uso del sistema. Unido a esto fueron descritos detalladamente los casos de uso del sistema para obtener así un mayor entendimiento de los requisitos funcionales previamente establecidos. Se especificó además la estructura del sistema, a través de los diagramas de clases del diseño, también las clases que ocupan un lugar relevante en el diseño, así como los patrones de diseño empleados para el proceso de desarrollo del plugin.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

Introducción

Basado en el análisis y diseño del sistema, en el presente capítulo se muestra el modelo de implementación que se ha realizado así como los diagramas de componentes y el de despliegue. Se describen las pruebas, con el objetivo de comprobar las funcionalidades del plugin en los diferentes escenarios.

3.1 Diagrama de Despliegue

El Modelo de Despliegue se utiliza para modelar el hardware utilizado en el desarrollo de los sistemas y las relaciones entre sus componentes. En muchos casos las vistas de despliegue implican la representación de la topología de hardware sobre el que se ejecuta el sistema.

Se presentan en el siguiente modelo de despliegue tres nodos que representan la computadora del usuario del sistema, esta se comunica con el servidor web apache donde está montado Redmine, interconectados por el protocolo HTTPS. El servidor web Apache se comunica con el servidor de base de datos PostgresSQL por el protocolo TCP/IP para obtener los datos necesarios.

Figura 13. Modelo de despliegue

3.2 Diagrama de Componentes

Dentro del Modelo de Implementación se encuentran los diagramas de componentes. Estos describen los elementos físicos y muestran una vista estática de un sistema, pueden estar compuestos por archivos, librerías, módulos, ejecutables, o paquetes y ser usados para modelar y documentar cualquier arquitectura del sistema. Permiten visualizar con más facilidad la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces.

El presente diagrama está compuesto por diferentes archivos de extensión erb o rb, que responden al funcionamiento del sistema del lado del cliente con el uso del framework Ruby on Rails, se representan físicamente cada una de las clases que se necesitan para el funcionamiento del plugin.

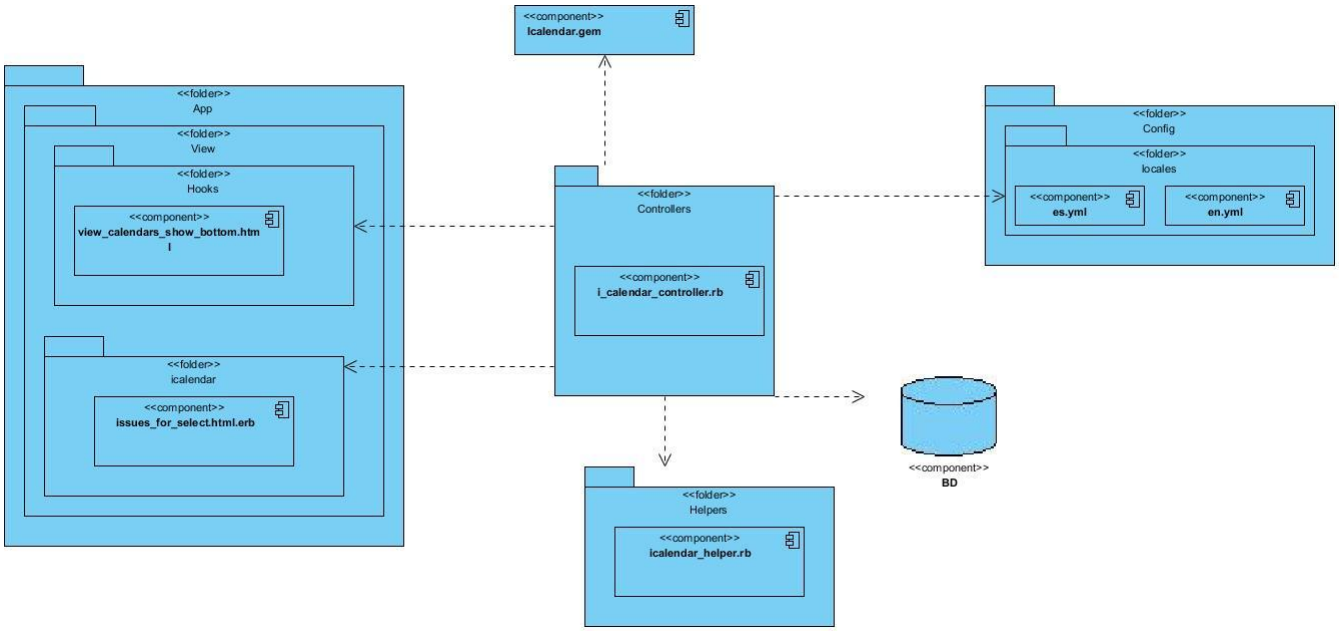


Figura 14. Diagrama de Componentes

Este diagrama está compuesto por 8 componentes, de los cuales son explicados algunos de ellos:

Fichero view_calendars_show_bottom.html.erb: E**s** un hook que se utiliza para insertar o modificar código y tiene como finalidad actualizar automáticamente en la plantilla show.html.erb del calendario el código definido en este fichero.

Fichero issues_for_select: Es el encargado de mostrar las tareas del calendario en una tabla que permita al usuario seleccionar las tareas que desea exportar en el calendario.

Fichero ICalendarController: Tiene la finalidad de manejar todos los eventos del sistema.

Librería Icalendar_gems: Es una gema o librería que se agrega en el Redmine, esta contiene un grupo de clases con funcionalidades del calendario. Se utiliza para redefinir la clase alarma del calendario dentro del método create_notification(cal,i,flag). En este método se definen los atributos de la notificación.

```
#creando la alarma en forma de notificacion por correo

alarm do
  action       "EMAIL"
  description  description_tmp # email body (required)
  summary      summary_tmp     # email subject (required)
  contacts.compact.uniq.each do |contact|
    add_attendee "mailto:#{contact.mail}"
  end
  trigger      trigger_mail_tmp
end
end
```

Figura 15. Definir alarma create_notification(cal,i,flag)

3.3 Estándar de codificación

Los estándares de codificación son modelos de programación que están orientados 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.

3.3.1 PascalCasing

Define que los nombres de los métodos, identificadores, clases y las variables, estén compuestos por una o más palabras juntas, iniciando cada palabra de las clases con letra mayúscula y el resto en minúscula. Este estándar se aplicó en toda la aplicación.

Nomenclatura de las clases según su tipo. Todas las clases del sistema están nombradas siguiendo el estándar PascalCasing, nombrándolas de acuerdo al propósito y la función de la misma.

3.3.2 CamelCasing

CamelCasing es equivalente a PascalCasing con la diferencia en la letra inicial del identificador que no comienza con mayúscula. Para definir el nombre de funciones y atributos del sistema se utilizó esta notación.

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

3.3.3 Notación húngara

La notación húngara se conoce también 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 el objetivo 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 4.Tipos 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.

Nomenclatura de las constantes. El identificativo de las constantes se realiza utilizando todas las letras en mayúscula.

A continuación se muestra un ejemplo de una clase del sistema donde se ven evidenciados los estándares de codificación.

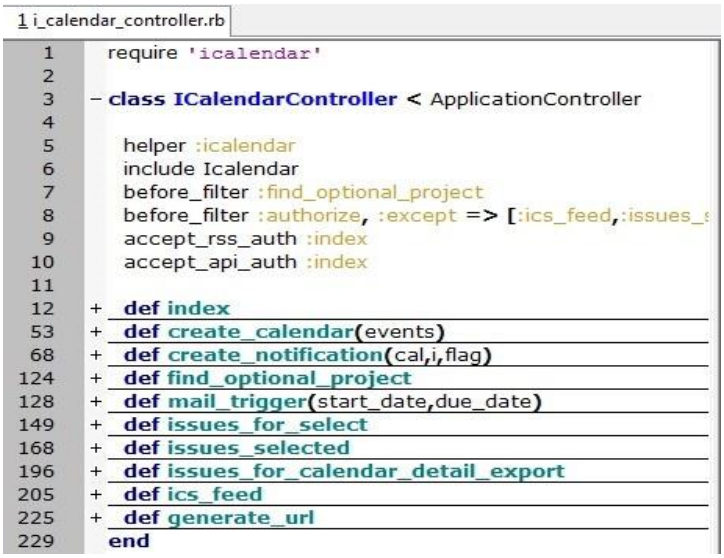


Figura 16. Estándar de codificación

3.4 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.

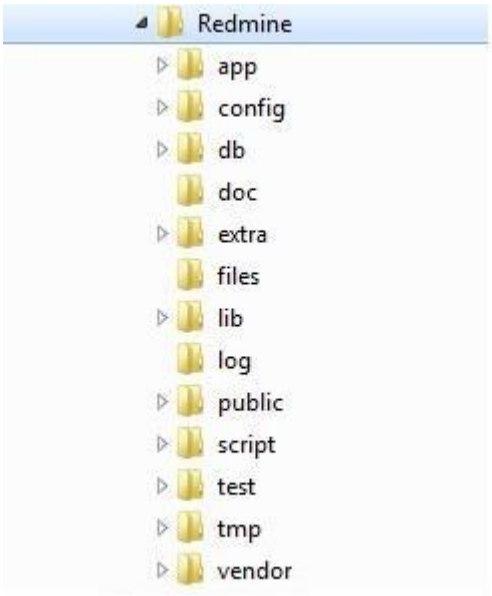


Figura 17.Estructura del marco de trabajo Redmine

Se centra la explicación del marco de trabajo en la carpeta **vendor** pues es en ella se va a trabajar en la creación del plugin.

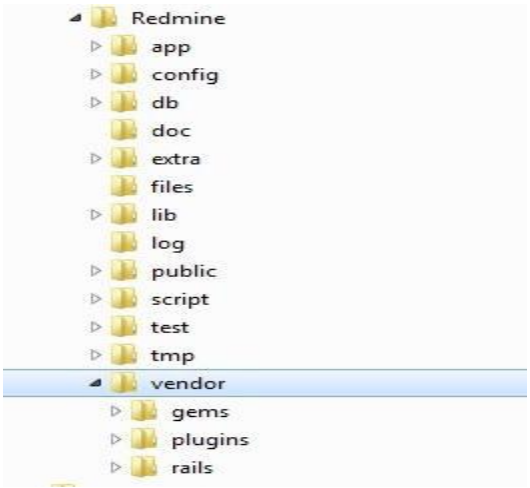


Figura 18.Marco de trabajo vendor

Dentro de la carpeta **vendor** se encuentran los componentes gems, rails, componentes vinculados al frameworks y plugins donde se crean y almacenan los plugins del Redmine. En la carpeta gems se agregó la gema Icalendar.

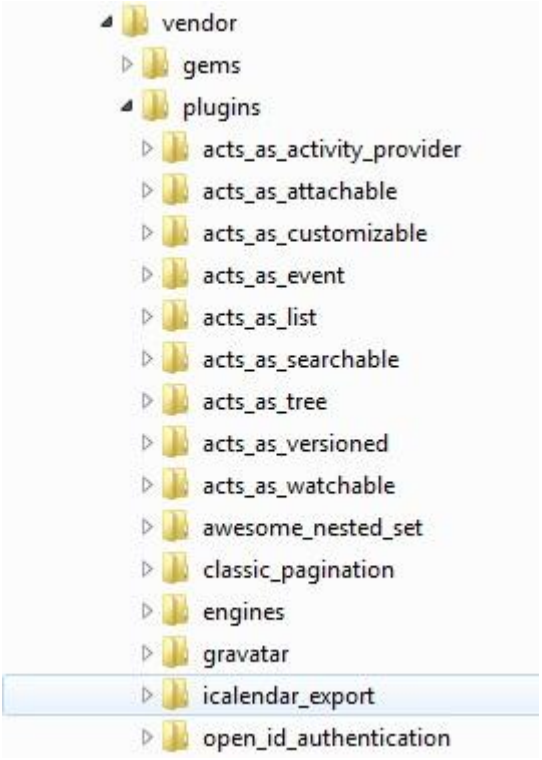


Figura 19.Macro de trabajo plugins

El nombre del plugin es **icalendar_export,** este contiene los elementos creados para su funcionamiento.

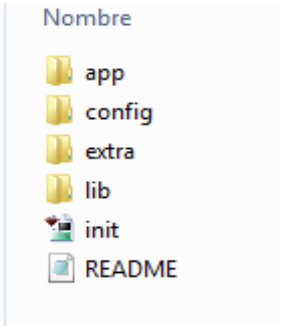


Figura 20. Marco de trabajo plugin redmine_importer

En la carpeta app se encuentran las principales clases del plugin, estas son:

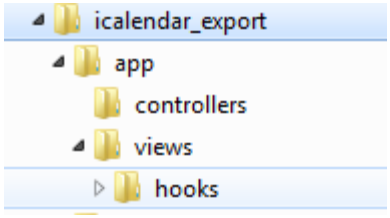


Figura 21. Marco de trabajo app

- **Controllers**: Contiene las clases controladoras. Donde se definen 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.
- **Views:** Contiene los ficheros que gestionan la capa de presentación, agrupados en la carpeta **hooks.**

3.5 Pruebas de Software

La prueba del software es un elemento crítico para la garantía de la calidad del software. El objetivo de la etapa de pruebas es garantizar la calidad del producto desarrollado. Las pruebas son un proceso que se enfoca sobre la lógica interna del software y las funciones externas, es un proceso de ejecución de un programa con la intención de descubrir un error. Una prueba tiene éxito si descubre un error no detectado hasta entonces. Además, esta etapa implica. (22)

- Verificar la integración adecuada de los componentes.
- Verificar que todos los requisitos se han implementado correctamente.
- Identificar y asegurar que los defectos encontrados se han corregido antes de entregar el software al cliente.

➢ **Nivel de Prueba**

Para comprobar que el plugin funcione de forma correcta, se realizan **Pruebas de Desarrollador** pues es el primer nivel de prueba y estas son diseñadas e implementadas por el equipo de desarrollo.

➢ **Técnica de Prueba**

Como técnica de prueba se aplicarán las **Pruebas de Funcionalidad** que son pruebas que se realizan fijando su atención en la validación de las funciones, métodos, servicios y caso de uso. Se analiza cada funcionalidad implementada para verificar que se cumplan todos los requisitos establecidos y de esta forma satisfacer las necesidades existentes.

➢ **Tipo de prueba**

Dentro de la pruebas de funcionalidad se realizan las **Pruebas Funcionales** que tienen como objetivo asegurar el trabajo apropiado de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados, se enfoca en los requisitos funcionales y casos de uso. El método que utiliza este tipo de prueba es el método de Caja Negra.

- **Método de prueba**

El método de prueba a utilizar es el método de **Caja Negra** que consiste en las pruebas que se llevan a cabo sobre la interfaz del software. También conocidas como Pruebas de Comportamiento, estas pruebas se basan en la especificación del programa o componente a ser probado para elaborar los casos de prueba. Las pruebas de Caja Negra pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto.
Para diseñar los casos de prueba de Caja Negra se utilizó la **Técnica de la Partición de Equivalencia** que divide el campo de entrada de un programa en variables de equivalencia con juegos de datos de entrada y salida. Las variables de equivalencia representan un conjunto de estados válidos y no válidos para las condiciones de entrada de un programa. Se definen dos tipos de variables de equivalencia, las *válidas*, que representan entradas válidas al programa, y las *no válidas*, que representan valores de entrada erróneos, aunque pueden existir valores no relevantes a los que no sea necesario proporcionar un valor real de dato.

3.6 Casos de prueba

Siempre que se desarrolla algún producto es necesario probar lo que se hizo para comprobar que cumple con todo lo que se precisó en el inicio de su elaboración; el software no está ajeno a las pruebas y son estas un elemento crítico para la garantía de la calidad. Se utilizó el método de caja negra que permite comprobar que cada funcionalidad es operativa. Las pruebas de Caja Negra se aplicarán mediante los Casos de Prueba que constituyen un conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para cumplir un objetivo en particular o una función esperada.

A continuación se presentan los casos de pruebas, para los casos de uso Exportar Calendario y Sincronizar Calendarios. Estas pruebas se realizan a través de una matriz de datos, para comprobar que el plugin funcione de forma correcta, donde:

V: indica válido, I: indica inválido, NA: que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.

- En la siguiente tabla se muestran las variables V1, V2, Vn…, que representan valores de entrada de datos para los casos de pruebas.

Descripción de las variables.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
		Exportar Calendario		
V1	% para avisar	listbox	Si	Opción que permite seleccionar el % de tiempo con que se desea que se cree la alarma, de no especificarse un valor se toma por defecto un valor predeterminado por el sistema.
V2	Marcar Tareas	checkbox	Si	Opción que inicialmente marca todas las tareas por defecto, quedando seleccionada una lista de tareas.

Tabla 5.Descripción de las variables

Caso de prueba para el CU- Exportar Calendario.

Escenario	Descripción	V1	V2	Respuesta del Sistema	Flujo Central
EC1. Exportar calendario a ICS.	En este escenario se exportarán las tareas a ICS por el usuario.	-	-	- Se obtendrá un fichero en formato ICS con todas las tareas exportadas.	1. Se da clic en el vínculo de Exportar ICS ubicado en la vista del calendario. 2. Seleccionar donde desea guardar el fichero de salida ICS. 3. Dar clic en el botón

					Guardar.
EC2. Exportar calendario detallado.	En este escenario se exportarán las tareas seleccionadas por el usuario a ICS con el tiempo de notificación que el mismo desee.	V	V	Se obtendrá un fichero en formato ICS con todas las tareas seleccionadas a exportar.	1. Se da clic en el vínculo de Exportar personalizado ubicado en la vista del calendario. 2. Se seleccionan las tareas que se deseen exportar. 3. Se especifica el tiempo de notificación para la alarma. 4. Se selecciona donde desea guardar el fichero de salida ICS. 3. Dar clic en el botón Guardar.

Tabla 6. Matriz de datos para el CU Exportar Calendario

Caso de prueba para el CU- Sincronizar calendario.

Escenario	Descripción	V1	V2	Respuesta del Sistema	Flujo Central

EC1. Sincronizar calendario.	En este escenario se definirá el URL para ser empleado en la sincronización de los calendarios.	-	-	-URL con la dirección del calendario para poder realizar la sincronización del mismo.	1. Se da clic en el vínculo de Generar URL. 2. Seleccionar donde desea guardar el fichero de salida con la dirección URL del calendario. 3. Dar clic en el botón Guardar.

Tabla 7.Matriz de datos para el CU Sincronizar Calendario

Después de realizar las pruebas de Caja Negra mediante los Casos de Prueba asociados a cada Caso de Uso, durante tres iteraciones y hasta no existir no conformidades pendientes, lo cual comprobó el correcto funcionamiento del plugin. Cada no conformidad detectada en el plugin fue registrada y clasificada para posteriormente ser solucionada; el registro de las mismas puede observarse a continuación junto a una tabla resumen.

PD: Pendiente **RA:** Resuelta

Fecha	Versión	Caso de Prueba	Cant. de No Conformidad	Cant. de No Conformidad PD	Cant. De No Conformidad RA
2/06/2012	1.0	CU-Exportar Calendario	3	-	3
2/06/2012	1.0	CU-Sincronizar Calendario	2	-	2

Tabla 8.Resumen de los resultado de las pruebas aplicadas

A continuación se muestra una tabla con las principales no conformidades detectadas durante el proceso de prueba.

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Clasificación	Estado NC	Respuesta del Equipo Desarrollo
Aplicación	1	En exportar personalizado no se registra el tiempo de notificación correctamente para cada tarea.	Exportar personalizado el calendario.	Prueba	S	1/06/12 PD 2/06/12 RA	Se corrigió el error, encontrado, ya las tareas exportadas contienen los tiempo de notificación de forma correcta.
Aplicación	2	No adiciona en la tabla de sincronización los id del los usuarios correspondiente a cada tarea	Exportar calendario.	Prueba	S	1/06/12 PD 2/06/12 RA	Se corrigió el error, cada tarea tiene su id de usuario correspon-diente.
Aplicación	3	Existen errores ortográficos en la vista de exportar personalizado. Palabra (Por ciento)	Exportar personalizado	Prueba	NS	1/06/12 PD 2/06/12 RA	Se corrigió el error, encontrado, la vista no presenta errores ortográficos. Palabra (Por ciento)
Aplicación	4	Existen errores ortográficos en	Sincronizar Calendarios	Prueba	NS	1/06/12 PD	Se corrigió el error,

		vínculo de Generar URL				2/06/12 RA	encontrado, la vista no presenta errores ortográficos. Palabra (Generar)
Aplicación	5	La URL conformada no contenía la llave del usuario válida.	Sincronizar Calendarios	Prueba	S	1/06/12 PD 2/06/12 RA	Se corrigió el error, encontrado, la URL posee la llave de acceso correcta para ese usuario.

Se evidencia a continuación los resultados de las no conformidades de las pruebas efectuadas en las tres iteraciones realizadas y divididas por tipos.

.

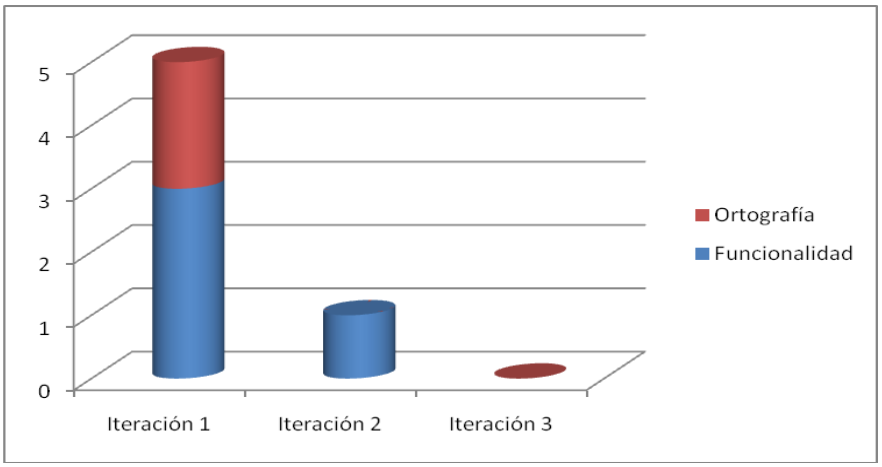


Figura 22. No conformidades por tipo e iteración

Se muestra a continuación los resultados de las clasificaciones de las no conformidades encontradas en la realización de los casos de pruebas por iteraciones.

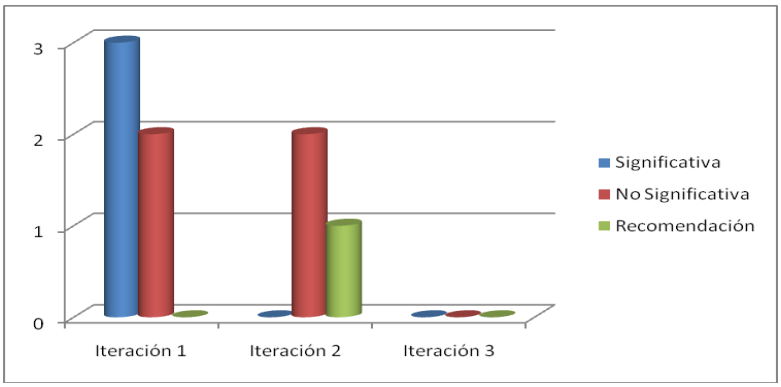


Figura 23. Clasificación de las No Conformidades por iteración

Conclusiones parciales

En este capítulo se realizó el modelo de implementación con el propósito de mostrar los componentes del sistema y sus relaciones, a través del diagrama de componentes. Además, se realizaron las pruebas para validar que los requisitos fueron implementados correctamente a través del método de caja negra, aplicando la técnica de partición de equivalencia, encontrándose un total de 5 no conformidades las cuales fueron solucionadas demostrando que el plugin realiza sus funcionalidades de forma correcta.

CONCLUSIONES

Con la realización del siguiente trabajo se logró resolver los problemas de interoperabilidad que existían entre el Redmine y la suite colaborativa Zimbra con 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, dando cumplimiento a los objetivos trazados para esta investigación.

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

- ✓ Se realizó un análisis de las prestaciones y características de la herramienta de gestión Redmine y la suite de colaboración Zimbra.
- ✓ Se definió el marco teórico de la investigación el cual ayudó a la compresión y desarrollo del sistema así como la elección de las herramientas usadas en el diseño e implementación.
- ✓ Se analizó y diseñó un plugin para la propuesta de integración entre Redmine y la suite de colaboración Zimbra.
- ✓ Se implementó el componente para la integración de las herramientas Redmine y Zimbra.
- ✓ Se realizó la validación la propuesta de integración empleando la técnica de prueba de Caja Negra.

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.
- Internacionalizar el plugin para su uso por toda la comunidad del Redmine de forma internacional.

REFERENCIAS BIBLIOGRÁFICAS

1. Centro de Excelencia de Software Libre. *Centro de Excelencia de Software Libre.* [En línea] [Citado el: 7 de Diciembre de 2011.] http://www.ceslcam.com/conocelo/analisis-de-aplicaciones/analisis/doc/analisis-de-aplicacion-Redmine/..

2. **proyectos, Redmine gestor de.** Tecnología Pyme. *Tecnología Pyme.* [En línea] 7 de Octubre de 2011. [Citado el: 17 de Noviembre de 2011.] http://www.tecnologiapyme.com/software/Redmine-gestor-de-proyectos-de-codigo-libre-para-nuestras-empresas.

3. **Peña, Rodrigo.** Gestión de proyecto. *Gestión de proyecto.* [En línea] Agosto de 2001. [Citado el: 7 de diciembre de 2011.] http://www.gestiopolis.com/recursos/documentos/fulldocs/ger/gestioproyecto.htm.

4. **Caro, Patricio Salinas.** Tutorial de UML. *Tutorial de UML.* [En línea] [Citado el: 4 de Diciembre de 2012.] http://www.dcc.uchile.cl/~psalinas/uml/introduccion.html.

5. **http://.** Gestion de Proyectos. Gestion de Proyectos. [En línea] [Citado el: 17 de Noviembre de 2011.] www.ciecem.uhu.es/centrodocumentacion/documentos/seminario/ponencias/GestionProyectosPepe_Pino.pdf.

6. Redmine. [En línea] [Citado el: 23 de Enero de 2012.] http://www.redmine.org/projects/redmine/wiki/Plugins.

7. VMware Zimbra. *VMware Zimbra.* [En línea] [Citado el: Noviembre de 26 de 2011.] http://www.Zimbra.com/buzz/index.es.html.

8. **Zamudio, Esmeralda Villegas y Méndez, Alejandra Virrueta.** *Investigación doumental. Metodologías de desarrollo de software.* Instituto tecnológico superior de Apatzingán, Michoacan. Apatzingán Michoacan : s.n., 2010.

9. OpenUP. *OpenUP.* [En línea] [Citado el: 6 de 1 de 2012.] http://epf.eclipse.org/wikis/openupsp/.

10. Free Download Manager. *Free Download Manager.* [En línea] 5 de Marzo de 2007. [Citado el: 12 de 12 de 2011.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/.

11. **Caro, Patricio Salinas.** Tutorial de UML. *Tutorial de UML.* [En línea] [Citado el: 4 de Diciembre de 2012.] http://www.dcc.uchile.cl/~psalinas/uml/introduccion.html.

12. NetBeans. *NetBeans.* [En línea] [Citado el: 14 de Diciembre de 2011.] http://netbeans.org/index_es.html.

13. Ruby on Rails. *Ruby on Rails.* [En línea] [Citado el: 26 de Noviembre de 2011.] http://rubyonrails.org.es/.

14. **Ruby.** Ruby. *Ruby.* [En línea] [Citado el: 14 de noviembre de 2011.] http://www.ruby-lang.org/es/about/.

15. Hosting PostgreSQL. [En línea] [Citado el: 20 de Enero de 2012.] http://www.hiperhosting.cl/hosting-postgresql.html.

16. Manual del usuario de PostgreSQL. [En línea] [Citado el: 16 de Enero de 2012.] http://mmc.igeofcu.unam.mx/LuCAS/Postgresql-es/web/navegable/user/x56.html.

17. **Gómez, Gloria Lucia Giraldo.** *Ingeniería de software clase 5, Actores y sus roles. Modelo de Dominio.* Escuela de sistemas, universidad nacional de Colombia, sede Merlin. Merlin, Colombia

18. Caso de Uso(Use Case). [En línea] [Citado el: 14 de Enero de 2012.] http://www.dcc.uchile.cl/~psalinas/uml/casosuso.html#casosuso.

19. proactiva-calidad. [En línea] [Citado el: 5 de Enero de 2012.] http://www.proactiva-calidad.com/java/patrones/mvc.html.

20. Estructura de las Aplicaciones Orientadas a Objetos. [En línea] [Citado el: 10 de Enero de 2012.] http://www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf.

21. asp. [En línea] [Citado el: 15 de Diciembre de 2012.] http://www.asp.net/mvc.

22. Pruebas-De-Software. [En línea] [Citado el: 12 de Enero de 2012.] http://www.buenastareas.com/ensayos/Pruebas-De-Software/1638252.html.

BIBLIOGRAFÍA

1. Centro de Excelencia de Software Libre. *Centro de Excelencia de Software Libre.* [En línea] [Citado el: 7 de Diciembre de 2011.] http://www.ceslcam.com/conocelo/analisis-de-aplicaciones/analisis/doc/analisis-de-aplicacion-Redmine/..

2. **proyectos, Redmine gestor de.** Tecnología Pyme. *Tecnología Pyme.* [En línea] 7 de Octubre de 2011. [Citado el: 17 de Noviembre de 2011.] http://www.tecnologiapyme.com/software/Redmine-gestor-de-proyectos-de-codigo-libre-para-nuestras-empresas.

3. **Peña, Rodrigo.** Gestión de proyecto. *Gestión de proyecto.* [En línea] Agosto de 2001. [Citado el: 7 de diciembre de 2011.] http://www.gestiopolis.com/recursos/documentos/fulldocs/ger/gestioproyecto.htm.

4. **Caro, Patricio Salinas.** Tutorial de UML. *Tutorial de UML.* [En línea] [Citado el: 4 de Diciembre de 2012.] http://www.dcc.uchile.cl/~psalinas/uml/introduccion.html.

5. **http://.** Gestion de Proyectos. Gestion de Proyectos. [En línea] [Citado el: 17 de Noviembre de 2011.] www.ciecem.uhu.es/centrodocumentacion/documentos/seminario/ponencias/GestionProyectosPepe_Pino.pdf.

6. Redmine. [En línea] [Citado el: 23 de Enero de 2012.] http://www.redmine.org/projects/redmine/wiki/Plugins.

7. VMware Zimbra. *VMware Zimbra.* [En línea] [Citado el: Noviembre de 26 de 2011.] http://www.Zimbra.com/buzz/index.es.html.

8. Zamudio, Esmeralda Villegas y Méndez, Alejandra Virrueta**.** *Investigación doumental. Metodologías de desarrollo de software.* Instituto tecnológico superior de Apatzingán, Michoacan. Apatzingán Michoacan : s.n., 2010.

9. OpenUP. *OpenUP.* [En línea] [Citado el: 6 de 1 de 2012.] http://epf.eclipse.org/wikis/openupsp/.

10. Free Download Manager. *Free Download Manager.* [En línea] 5 de Marzo de 2007. [Citado el: 12 de 12 de 2011.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/.

11. Caro, Patricio Salinas**.** Tutorial de UML. *Tutorial de UML.* [En línea] [Citado el: 4 de Diciembre de 2012.] http://www.dcc.uchile.cl/~psalinas/uml/introduccion.html.

12. NetBeans. *NetBeans.* [En línea] [Citado el: 14 de Diciembre de 2011.] http://netbeans.org/index_es.html.

13. Ruby on Rails. *Ruby on Rails.* [En línea] [Citado el: 26 de Noviembre de 2011.] http://rubyonrails.org.es/.

14. **Ruby.** Ruby. *Ruby.* [En línea] [Citado el: 14 de noviembre de 2011.] http://www.ruby-lang.org/es/about/.

15. Hosting PostgreSQL. [En línea] [Citado el: 20 de Enero de 2012.] http://www.hiperhosting.cl/hosting-postgresql.html.

16. Manual del usuario de PostgreSQL. [En línea] [Citado el: 16 de Enero de 2012.] http://mmc.igeofcu.unam.mx/LuCAS/Postgresql-es/web/navegable/user/x56.html.

17. **Gómez, Gloria Lucia Giraldo.** *Ingeniería de software clase 5, Actores y sus roles. Modelo de Dominio.* Escuela de sistemas, universidad nacional de Colombia, sede Merlin. Merlin, Colombia

18. Caso de Uso(Use Case). [En línea] [Citado el: 14 de Enero de 2012.] http://www.dcc.uchile.cl/~psalinas/uml/casosuso.html#casosuso.

19. proactiva-calidad. [En línea] [Citado el: 5 de Enero de 2012.] http://www.proactiva-calidad.com/java/patrones/mvc.html.

20. Estructura de las Aplicaciones Orientadas a Objetos. [En línea] [Citado el: 10 de Enero de 2012.] http://www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf.

21. asp. [En línea] [Citado el: 15 de Diciembre de 2012.] http://www.asp.net/mvc.

22. Pruebas-De-Software. [En línea] [Citado el: 12 de Enero de 2012.] http://www.buenastareas.com/ensayos/Pruebas-De-Software/1638252.html.

23. Rails [En línea] [Citado el: 16 de Marzo de 2012.] http://rubyonrails.org/

24. Welcome to Rails Docs [Citado el: 5 de abril de 2012.] http://www.api.rubyonrails.org/

25. Welcome to Rails guides [Citado el: 5 de abril de 2012.] http://guides.rubyonrails.org/

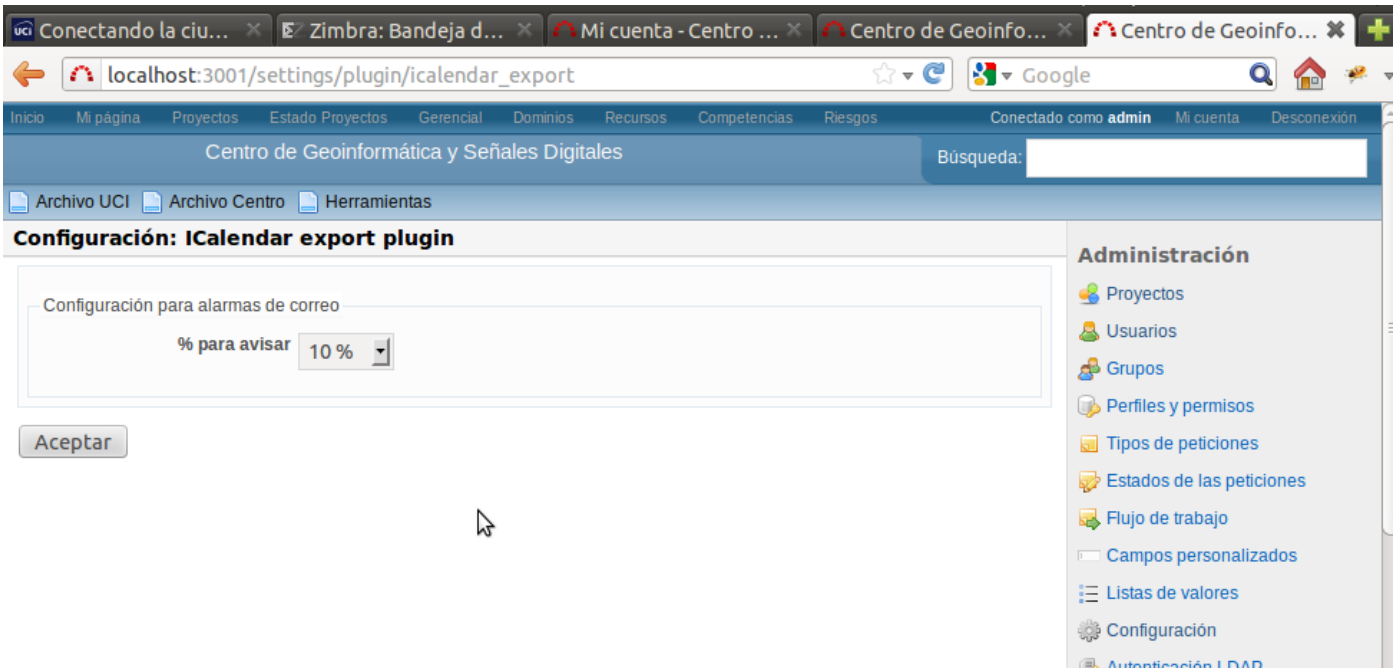
26. Redmine, gestor de proyectos de código libre para nuestras empresas [Citado el: 5 de abril de 2012.] http://www.tecnologiapyme.com/software/

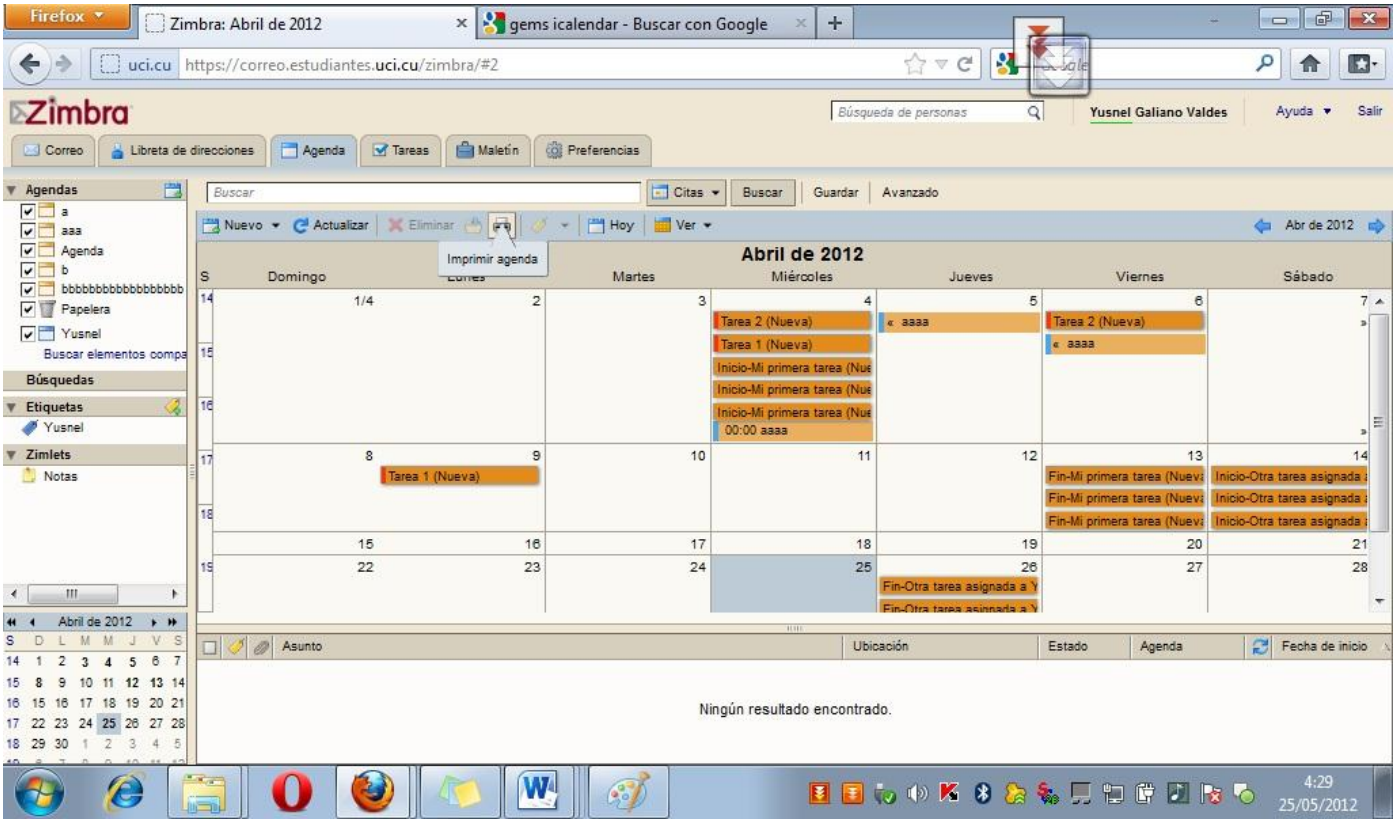
27. Redmine Integrated SCM & Project Management [Citado el: 5 de abril de 2012.] http://www.turnkeylinux.org/redmine

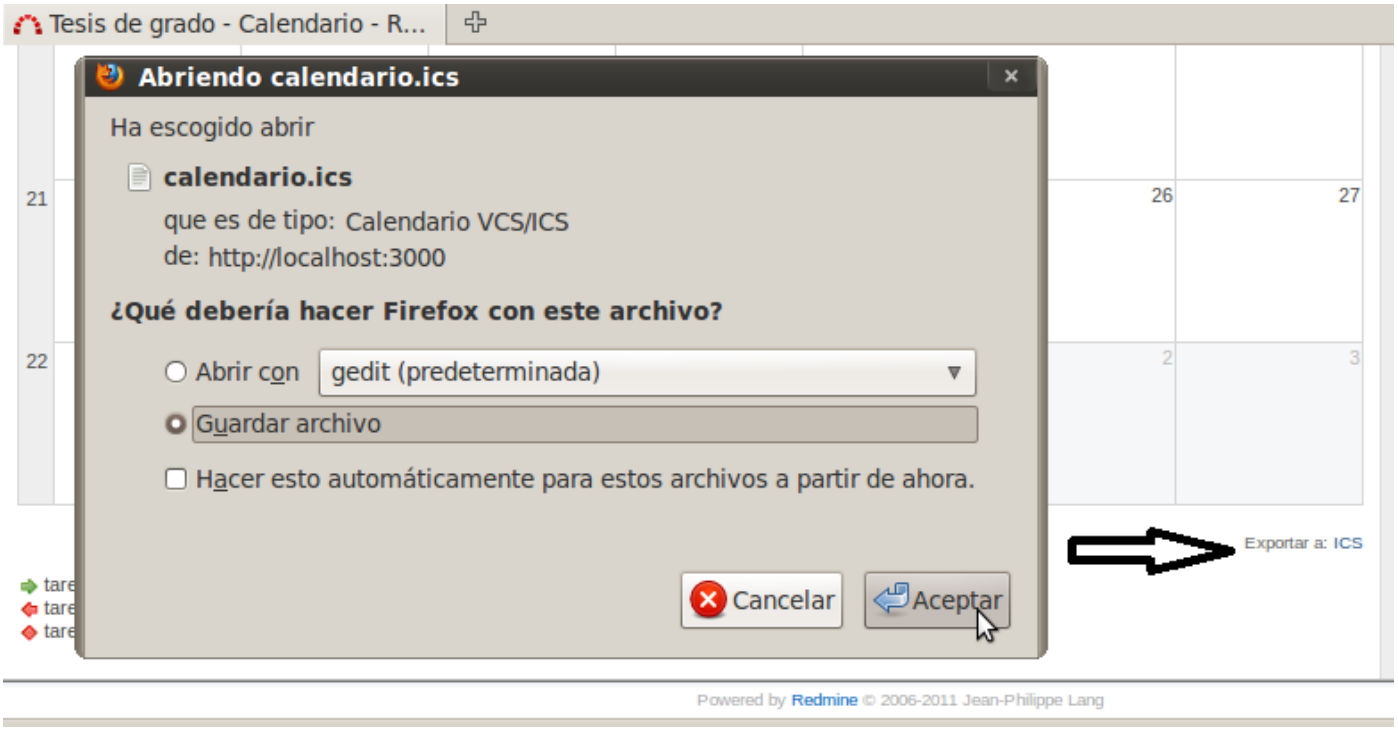
28. Redmine y zimbra [Citado el: 5 de abril de 2012.] http://www.tommyblue.it/2010/09/01/accedere-a-redmine-con-le-credenziali-di-zimbra

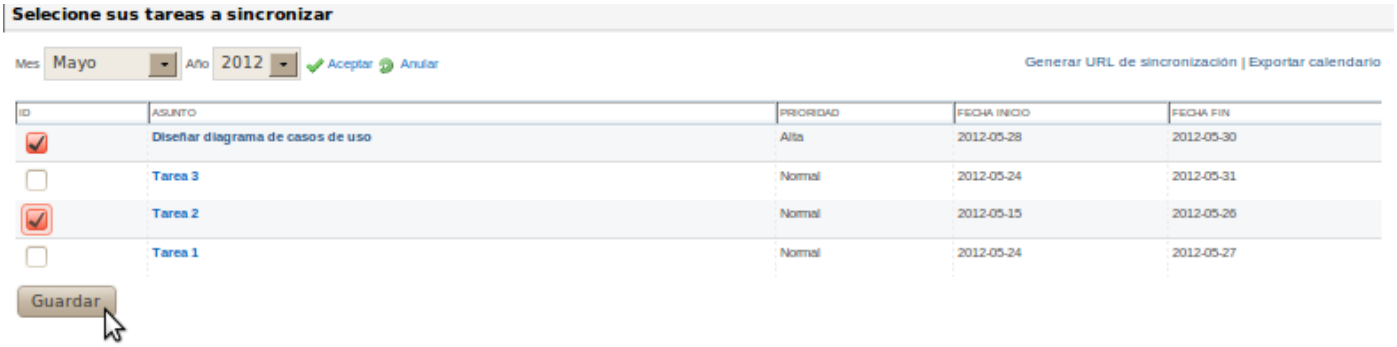
ANEXOS

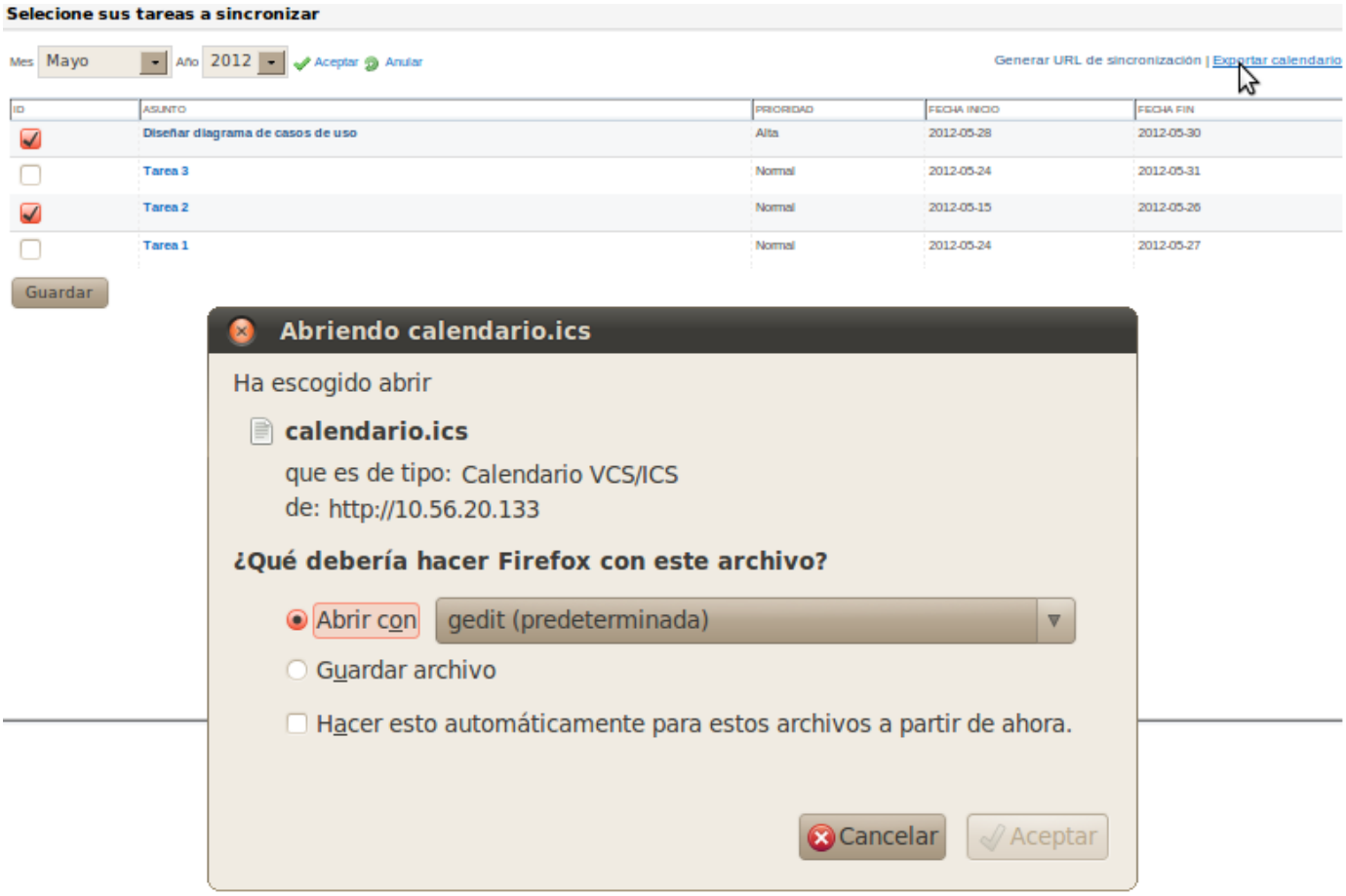
Anexo 1. Imágenes de la Aplicación

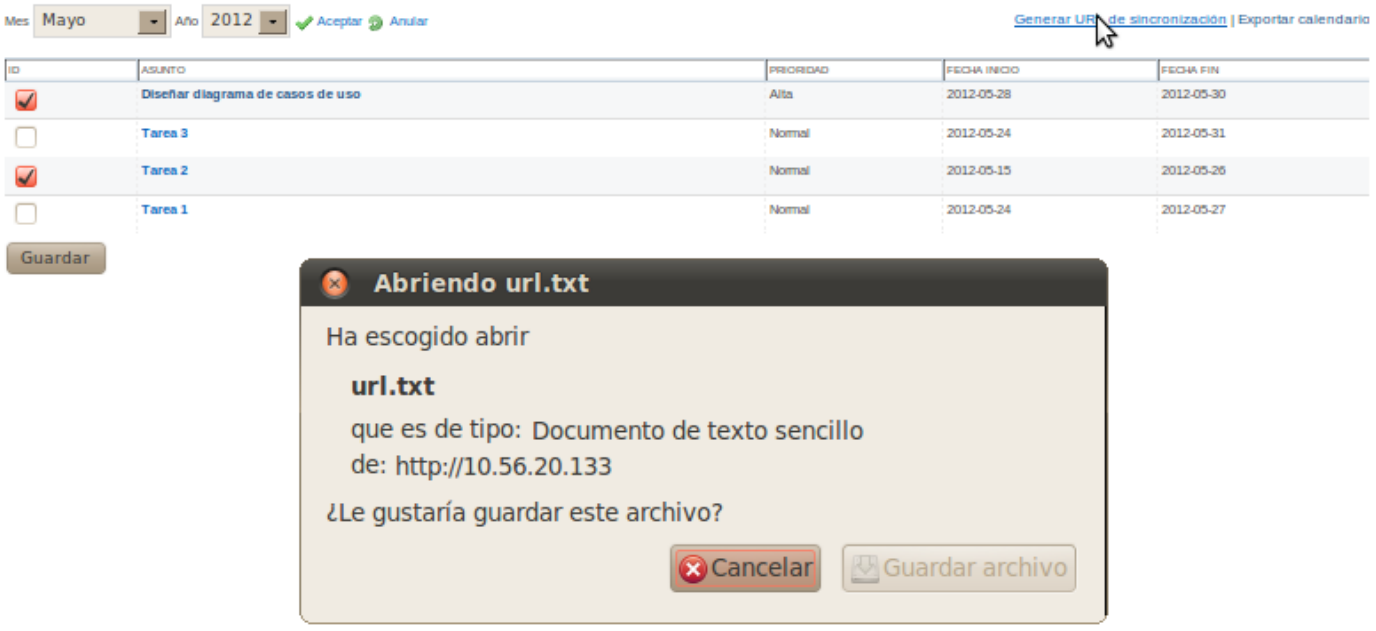


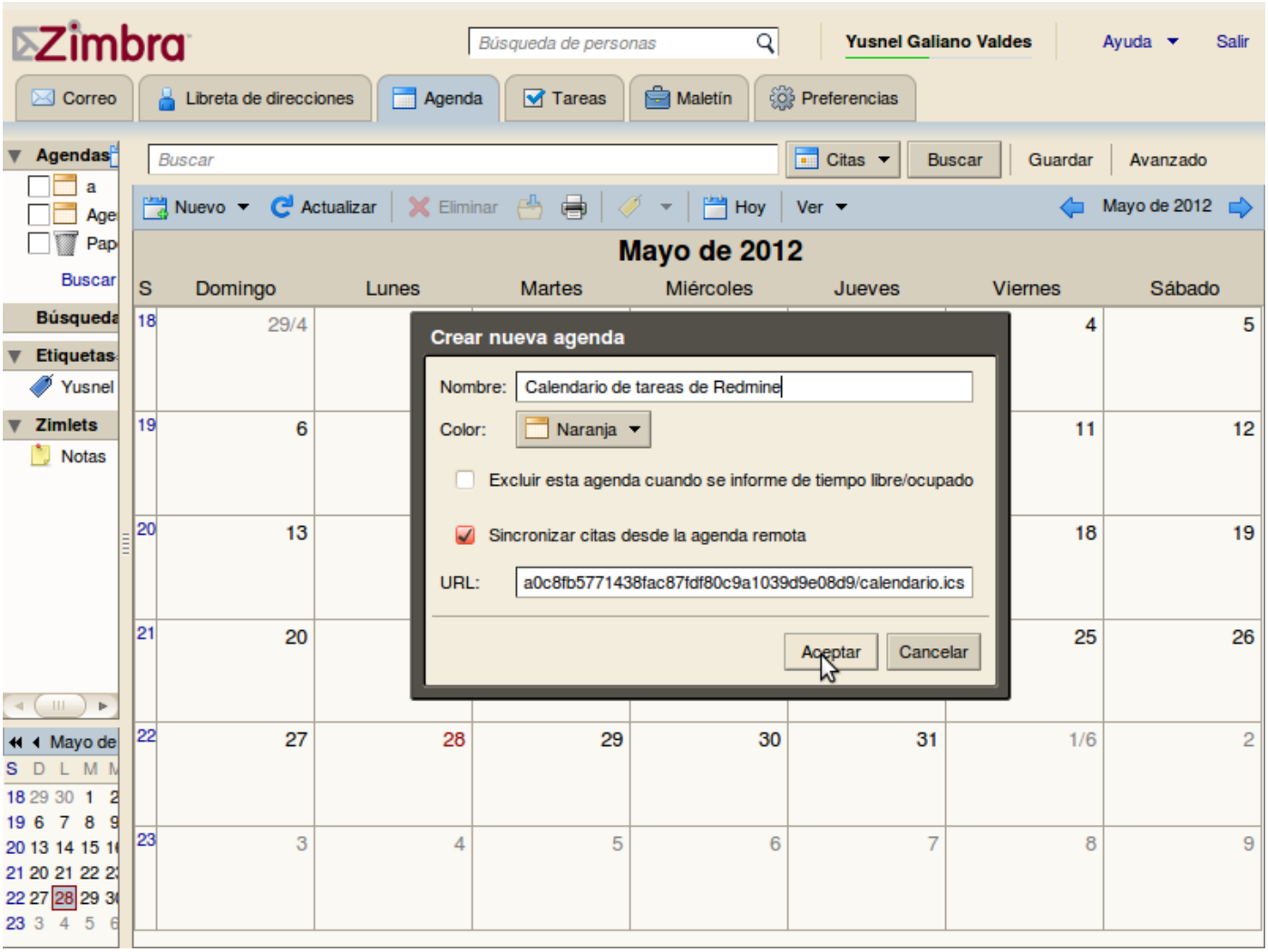

Firefox
Zimbra: Abril de 2012
gems icalendar - Buscar con Google
uci.cu https://correo.estudiantes.uci.cu/zimbra/#2
Zimbra
Búsqueda de personas
Yusnel Galiano Valdes
Ayuda
Salir
Correo
Libreta de direcciones
Agenda
Tareas
Maletín
Preferencias
Agendas
a
aaa
Agenda
b
bbbbbbbbbbbbbbbbb
Papelera
Yusnel
Buscar elementos compa
Búsquedas
Etiquetas
Yusnel
Zimlets
Notas
Buscar
Citas
Buscar
Guardar
Avanzado
Nuevo
Actualizar
Eliminar
Imprimir agenda
Hoy
Ver
Abr de 2012
Abril de 2012
S
Domingo
Lunes
Martes
Miércoles
Jueves
Viernes
Sábado
Tarea 2 (Nueva)
Tarea 1 (Nueva)
Inicio-Mi primera tarea (Nue
Inicio-Mi primera tarea (Nue
Inicio-Mi primera tarea (Nue
00:00 aaaa
« aaaa
Tarea 2 (Nueva)
« aaaa
Tarea 1 (Nueva)
Fin-Mi primera tarea (Nueva
Fin-Mi primera tarea (Nueva
Fin-Mi primera tarea (Nueva
Inicio-Otra tarea asignada a
Inicio-Otra tarea asignada a
Inicio-Otra tarea asignada a
Fin-Otra tarea asignada a Y
Asunto
Ubicación
Estado
Agenda
Fecha de inicio
Ningún resultado encontrado.
4:29
25/05/2012


Tesis de grado - Calendario - R...
Abriendo calendario.ics
Ha escogido abrir
calendario.ics
que es de tipo: Calendario VCS/ICS
de: http://localhost:3000
¿Qué debería hacer Firefox con este archivo?
Abrir con
gedit (predeterminada)
Guardar archivo
Hacer esto automáticamente para estos archivos a partir de ahora.
Cancelar
Aceptar
Exportar a: ICS
Powered by Redmine © 2006-2011 Jean-Philippe Lang


Selecione sus tareas a sincronizar
Mes Mayo
Año 2012
Aceptar
Anular
Generar URL de sincronización | Exportar calendario
ID
ASUNTO
PRIORIDAD
FECHA INICIO
FECHA FIN
Diseñar diagrama de casos de uso
Alta
2012-05-28
2012-05-30
Tarea 3
Normal
2012-05-24
2012-05-31
Tarea 2
Normal
2012-05-15
2012-05-26
Tarea 1
Normal
2012-05-24
2012-05-27
Guardar


Selecione sus tareas a sincronizar
Mes Mayo Año 2012 Aceptar Anular
Generar URL de sincronización | Exportar calendario
ID ASUNTO PRIORIDAD FECHA INICIO FECHA FIN
Diseñar diagrama de casos de uso Alta 2012-05-28 2012-05-30
Tarea 3 Normal 2012-05-24 2012-05-31
Tarea 2 Normal 2012-05-15 2012-05-26
Tarea 1 Normal 2012-05-24 2012-05-27
Guardar
Abriendo calendario.ics
Ha escogido abrir
calendario.ics
que es de tipo: Calendario VCS/ICS
de: http://10.56.20.133
¿Qué debería hacer Firefox con este archivo?
Abrir con
gedit (predeterminada)
Guardar archivo
Hacer esto automáticamente para estos archivos a partir de ahora.
Cancelar
Aceptar


Mes Mayo Año 2012 Aceptar Anular
Generar URL de sincronización | Exportar calendario
ID ASUNTO PRIORIDAD FECHA INICIO FECHA FIN
Diseñar diagrama de casos de uso Alta 2012-05-28 2012-05-30
Tarea 3 Normal 2012-05-24 2012-05-31
Tarea 2 Normal 2012-05-15 2012-05-26
Tarea 1 Normal 2012-05-24 2012-05-27
Guardar
Abriendo url.txt
Ha escogido abrir
url.txt
que es de tipo: Documento de texto sencillo
de: http://10.56.20.133
¿Le gustaría guardar este archivo?
Cancelar
Guardar archivo


Zimbra
Búsqueda de personas
Yusnel Galiano Valdes
Ayuda
Salir
Correo
Libreta de direcciones
Agenda
Tareas
Maletín
Preferencias
Agendas
Buscar
Citas
Buscar
Guardar
Avanzado
Nuevo
Actualizar
Eliminar
Hoy
Ver
Mayo de 2012
Mayo de 2012
Etiquetas
Yusnel
Zimlets
Notas
Crear nueva agenda
Nombre: Calendario de tareas de Redmine
Color: Naranja
Excluir esta agenda cuando se informe de tiempo libre/ocupado
Sincronizar citas desde la agenda remota
URL: a0c8fb5771438fac87fdf80c9a1039d9e08d9/calendario.ics
Aceptar
Cancelar

GLOSARIO DE TÉRMINOS

A

API: Una interfaz de programación de aplicaciones o API (del inglés Application Programming Interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

F

Framework: Es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado.

G

Gems: Las gemas son conocidas como plugins o librerías que permiten utilizar código ya definido.

H

Herramienta CASE: (Computer Aided/Assisted Software/System Engineering) o (Ingeniería de Software Asistida por Computadora) son herramientas utilizadas en el modelamiento visual de sistemas.

Hook: Es un gancho, usado para insertar o modificar códigos en plantillas sin tener que editar los archivos de plantilla.

I

IDE: Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

ICS: Es un tipo de archivo para el formato de calendarios.

ICalendar: Es un estándar para el intercambio de información de calendarios. El estándar también se conoce como "iCal".

O

OpenUP: Proceso unificado abierto, constituye una metodología de código abierto, que mantiene las características esenciales de la metodología RUP. Está desarrollado para pequeños equipos organizados.

P

Plug-in: Es Una Aplicación Informática que Interactúa Con Otra Aplicación Para Aportarle Una Función O Utilidad Específica, Generalmente Muy Específica, Como Por Ejemplo Servir Como Driver En Una Aplicación, Para Hacer Así Funcionar Un Dispositivo En Otro Programa.

R

RUP: El Proceso Unificado Racional o de Desarrollo (Rational Unified Process en inglés, tradicionalmente abreviado como RUP) constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

U

UML: Un lenguaje para modelar objetos es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar parte de un diseño de software orientado a objetos.