4. Resultados

4.1. Análisis de la arquitectura

Como ya se ha dicho anteriormente, Project.net es una aplicación web que corre sobre Apache Tomcat. Cuando se utiliza para ejecutar aplicaciones Web, Tomcat puede servir páginas HTML, sin configuración adicional, ya que está configurado con un conector HTTP que puede procesar solicitudes del navegador Web de un usuario. Debido a este conector, Tomcat puede funcionar como servidor Web independiente. Puede servir páginas HTML estáticas y también procesar páginas JSP y Servlet.
Los conectores de Tomcat constituyen la interfaz externa (sobre HTTP o HTTPS) de los clientes Tomcat. Existen dos tipos de conectores: los que implementan una pila HTTP propia (denominados conectores HTTP) y los que vinculan Tomcat a un servidor Web externo como Apache o IIS (denominados conectores de servidor Web). En nuestro caso, Tomcat funcionará de forma autónoma, por lo que, nos centraremos en los conectores HTTP puesto que son los que utilizamos para hacer funcionar Project.net.
En los anexos se detalla la configuración de estos conectores [15].
Resumiendo, Apache Tomcat está a la espera de solicitudes HTTP en el puerto 7070, se encarga de aceptar las solicitudes, dirigirlas a la aplicación Web concreta y al recurso específico, y de devolver el resultado del procesamiento de la solicitud.
Las solicitudes se redirigirán al servlet predeterminado en función del patrón de su URL.
Las direcciones URL que se escriban en el navegador y que terminen en .jsp o .jspx, ésta se pasará a través de un compilador JSP que compilará el archivo en Java, y tras ello, se compilará el archivo Java en una clase Servlet, el resto se pasarán al servlet predeterminado. Seguidamente, se ejecutará esta clase y el resultado se mostrará al cliente en su navegador.
En el lado del cliente sólo existirá un navegador, por lo que todos los recursos se consumirán en la maquina servidor.
Para guardar los datos del cada unos de los proyectos para los que se utilice Project.net, Apache Tomcat deberá tener conexión a una base de datos.
Además, como los participantes de los proyectos reciben los avisos y notificaciones mediante correo electrónico, Tomcat también deberá estar conectado a un servidor de correo.

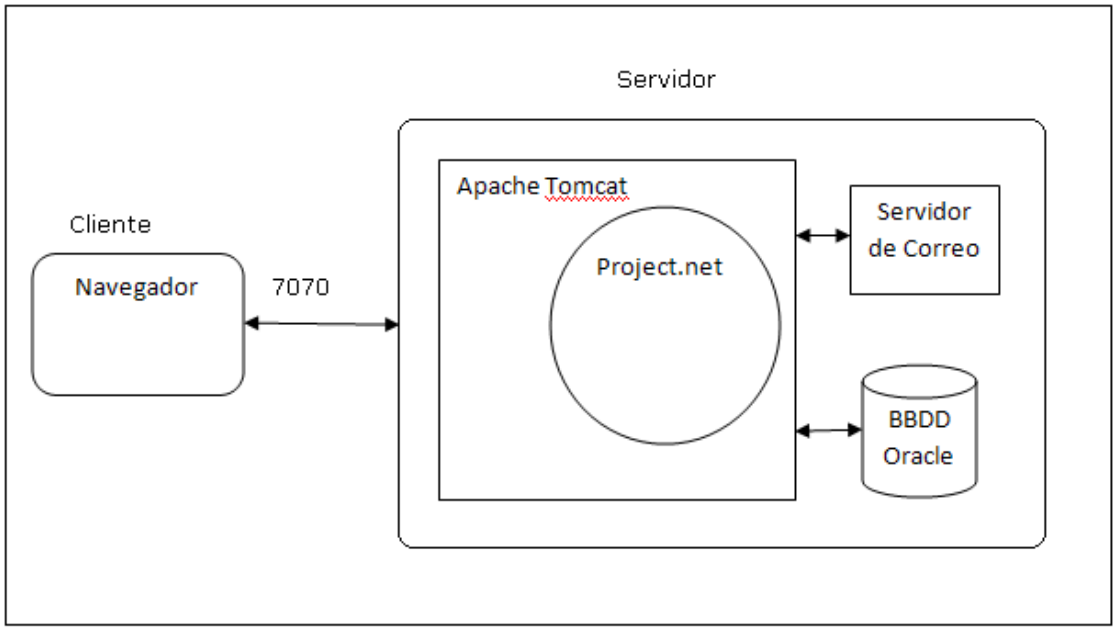


Ilustración 3: Diagrama de bloques de la arquitectura de Project.net

El servidor de correo puede estar tanto en la misma máquina donde se instala la aplicación como en el exterior.

4.2. Análisis funcional

Project.net está pensado para maximizar el rendimiento de cualquier organización en la planificación o seguimiento de un proyecto o conjunto de ellos, proporcionando información en tiempo real sobre el estado de los mismos [1].
Permite a cada uno de los miembros del equipo, la gestión de sus tareas individuales, así como la visualización de ellas dentro del proyecto global.
Se detallará una por una, las funcionalidades más relevantes de Project.net que se encuentra un miembro de un proyecto al acceder a Project.net [4].

- Personal Dashboards

Proporciona una vista personalizada de los proyectos, tareas, progreso del trabajo realizado y documentos. Sólo es visible por uno mismo, y permite un fácil acceso a:

- A los proyectos a los que se es miembro.
- Las tareas que se le asignan.
- Calendario y blog.
- Próximas reuniones.

- Documentos accesibles. En Project.net se pueden subir documentos, los cuales podrán ser descargados por cualquier miembro que pertenezca al proyecto.

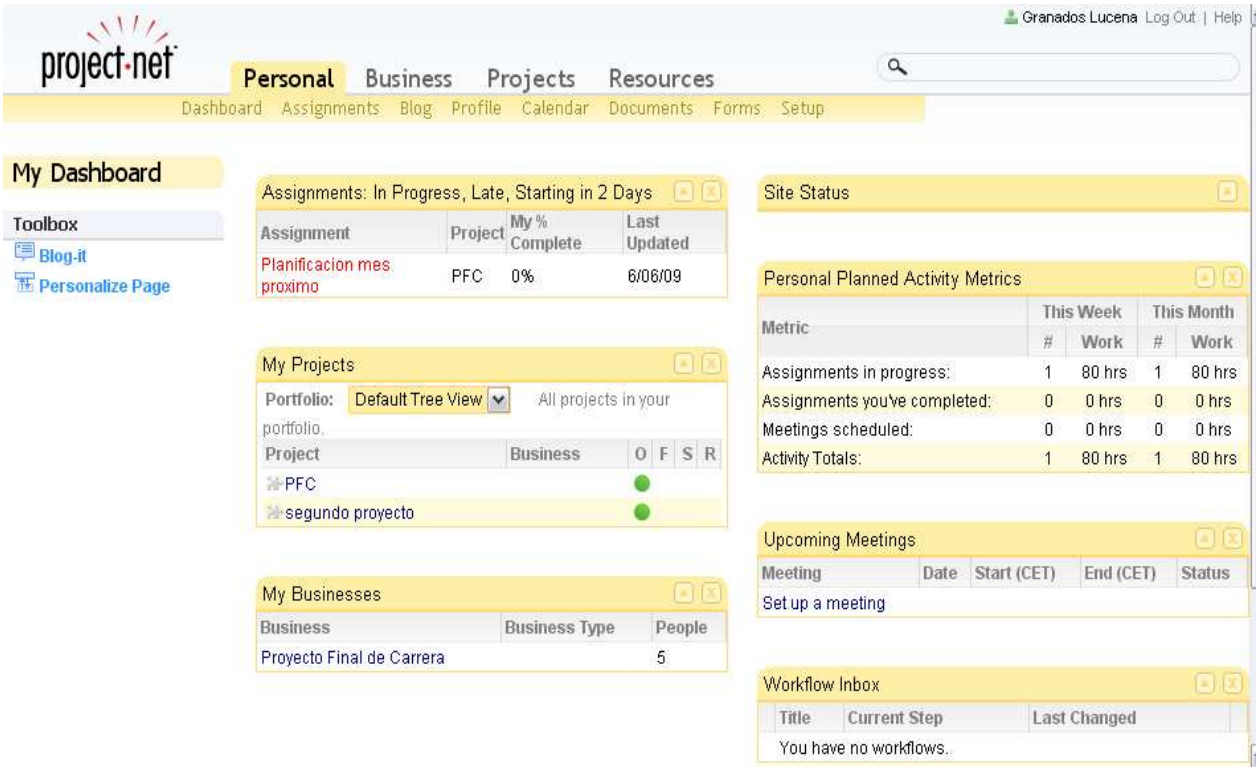


Ilustración 4: Pantalla inicial tras el Login

- Assignments

Permite revisar todas las tareas que le han sido asignadas, el plazo que tienen y su progreso, y filtrarlas según el proyecto al que pertenezcan.

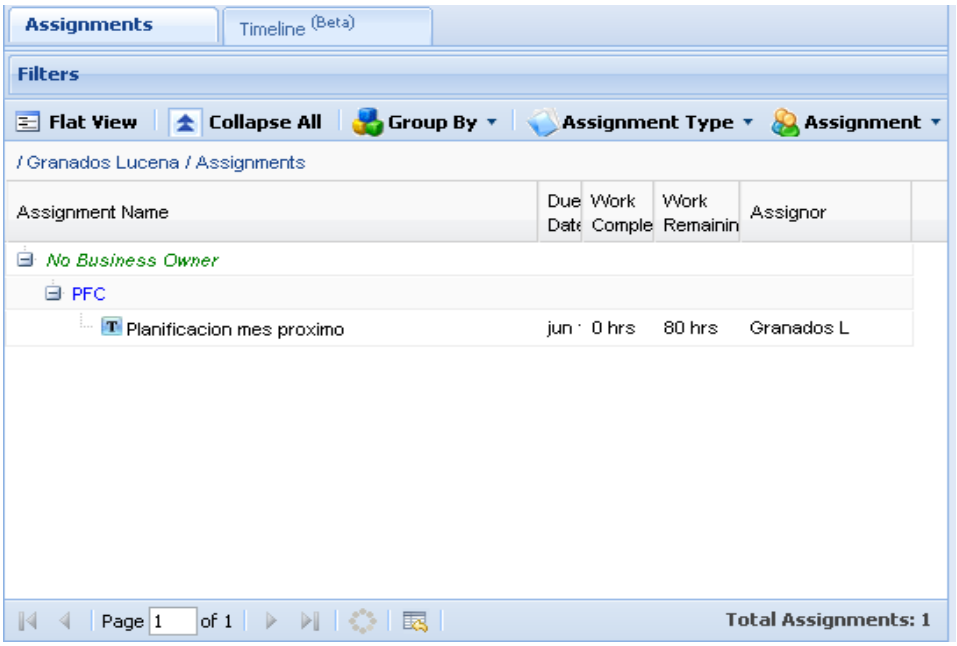


Ilustración 5: Visualización tareas asignadas

- Blog

Project.net facilita la comunicación entre los distintos miembros mediante blogs.

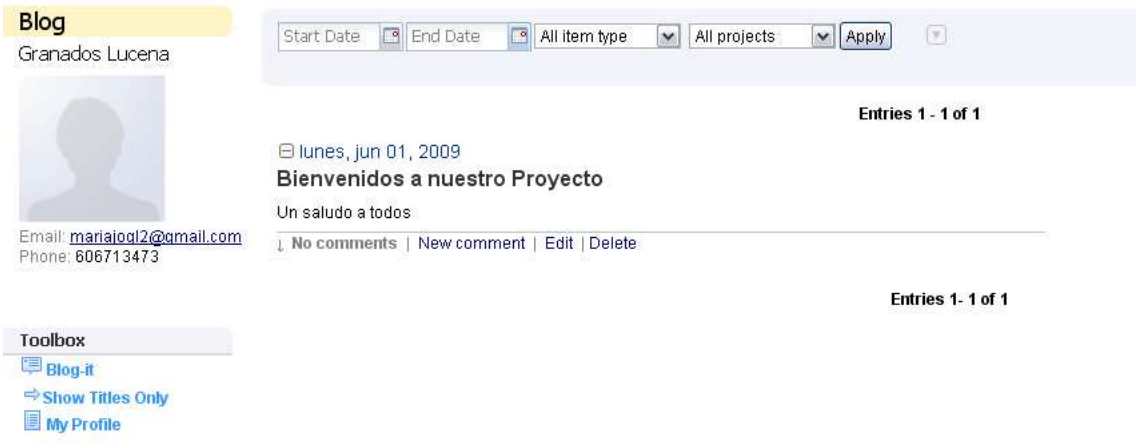


Ilustración 6: Visualización del blog relativo a un proyecto

- Profile

Permite almacenar los datos personales e información de contacto de cada uno de los miembros del proyecto.

- Project Dashboards

A diferencia de Personal Dashboards, éste es visible por todos los miembros pertenecientes a un proyecto. Se proporciona toda la información de un proyecto, incluyendo hitos, tareas, documentos, resultados, problemas, miembros.

Ilustración 7: Pantalla inicial de un proyecto

- Wiki

Con los wikis su equipo puede:

- Compartir información e ideas.
- Fomentar la colaboración.
- Facilitar la revisión y la actualización.
- Seguimiento del historial del proyecto.

- Workplan

El plan de trabajo muestra las personas a las que se les asigna cada tarea. Con esta aplicación, los jefes de proyecto podrán:

- Ver todas las tareas actuales, y las etapas en las que se encuentran. Así como, ir modificando el progreso de éstas, conforme se haya realizado el trabajo correspondiente.
- Programar más tareas y asignárselas a la persona correspondiente.
- Modificar datos, integrándolos con los blogs o wikis.
- Estimar tiempo para completar el trabajo asignado.
- Dividir el proyecto en fases.
- Cargar y guardar archivos de Microsoft Project.

El resto de miembros de un proyecto también podrán ver todas las tareas del proyecto, y las personas a las que están asignadas, pero sólo podrán editar la tarea asignados a ellos mismos. Tampoco podrán crear tareas nuevas ni fases.

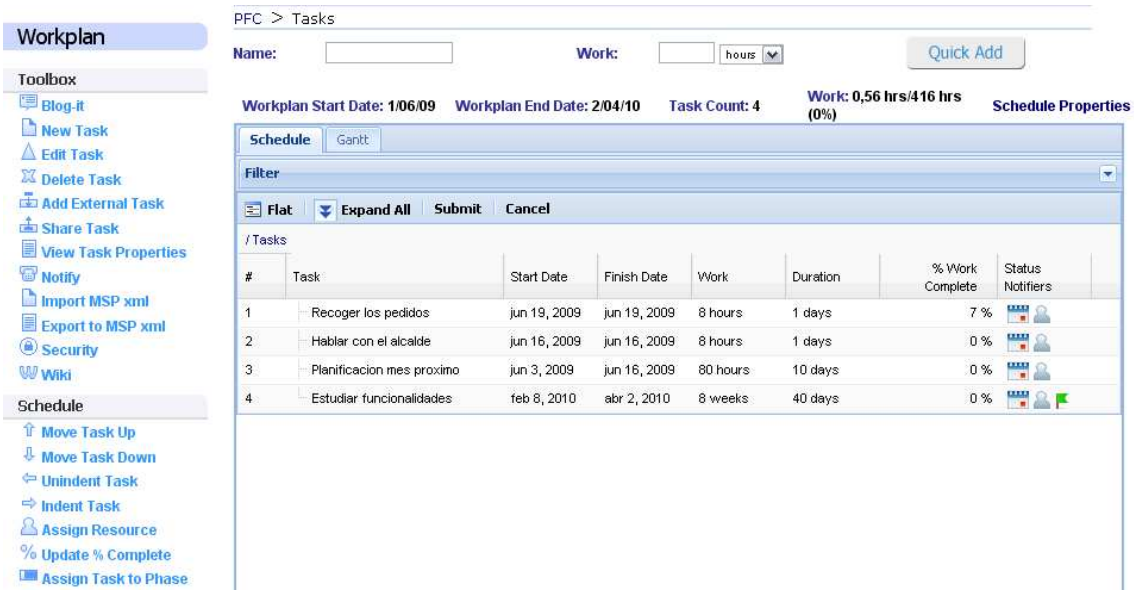


Ilustración 8: Visualización de tareas y personas a las que están asignadas

A partir de las tareas existentes, automáticamente se genera un diagrama de Gantt.

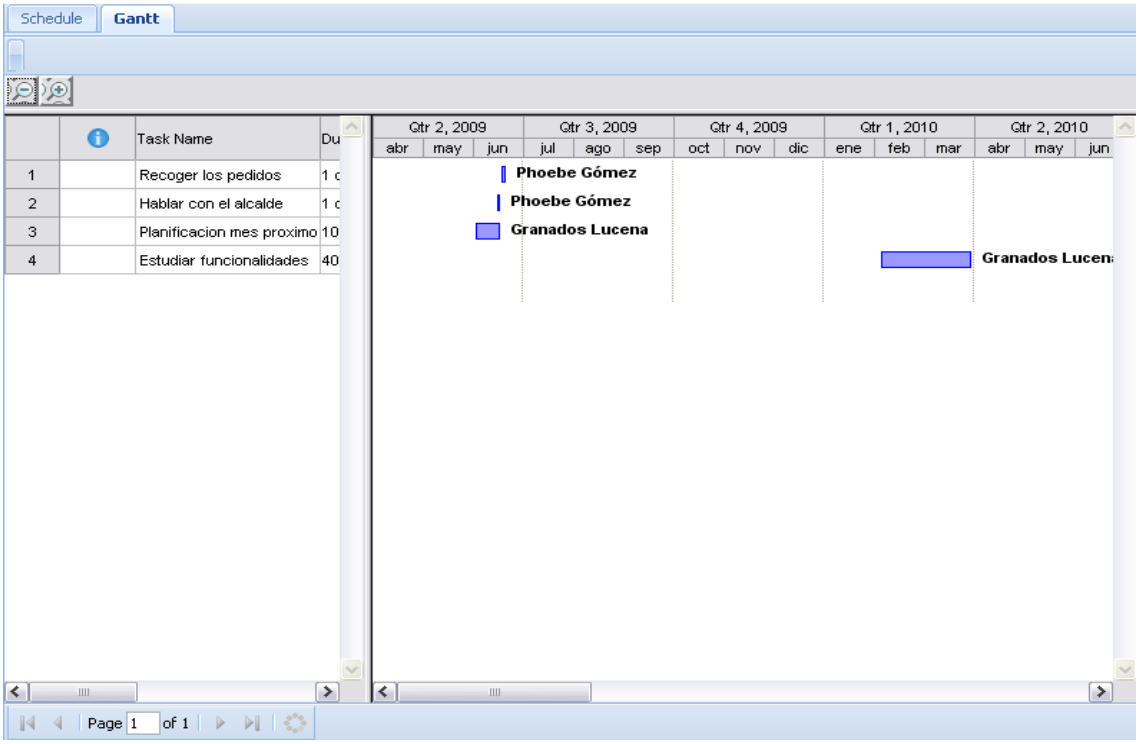


Ilustración 9: Visualización Diagrama de Gantt de un proyecto

- *Workflow*

En Project.net se pueden crear flujos de trabajo, con éstos se estructuran las tareas, se explica cómo se realizan, cuál es su orden correlativo, como se

sincronizan, como fluye la información que soporta las tareas, y cómo se le hace el seguimiento a las tareas.

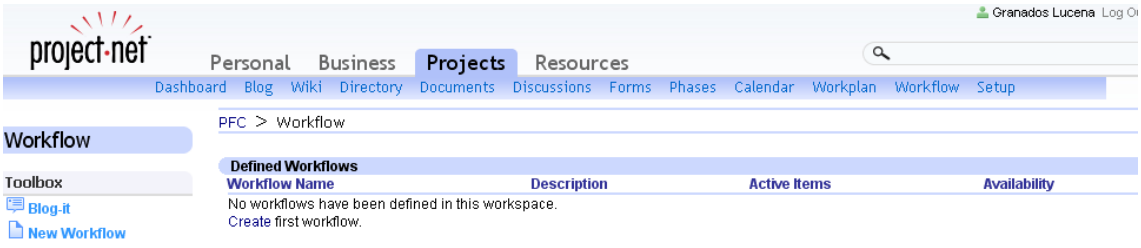


Ilustración 10: Visualización Workflows

- Phases

Esta herramienta de Project.net ayudará a formalizar la metodología seguida para el desarrollo de un proyecto, resultando bastante útil para proyectos grandes. Se podrán crear distintas fases. Las diferentes fases de un proyecto constituyen el ciclo de vida de éste. Una fase incluye una serie de tareas, y la decisión de pasar de fase puede basarse en si todas las tareas se han completado. Esta es la forma de ir avanzando en las diferentes etapas de un proyecto.

- Forms

Con la herramienta Forms se pueden crear diferentes formularios. Una vez creados, se guardarán en la biblioteca de formularios, que servirán a la organización para estandarizar datos sobre cuestiones de diverso tipo.

- Project Templates

Project.net permite almacenar plantillas de trabajo, para posteriormente utilizarlos en proyectos similares, con el consecuente ahorro de tiempo. Se incluirán flujos de trabajo, formularios, documentos, grupos de discusión, resultados y la documentación del usuario.

- Calendar

Dentro de cada proyecto existe una herramienta Calendar, donde se puede visualizar las tareas de forma muy fácil, y agregar nuevas tareas, eventos o reuniones.

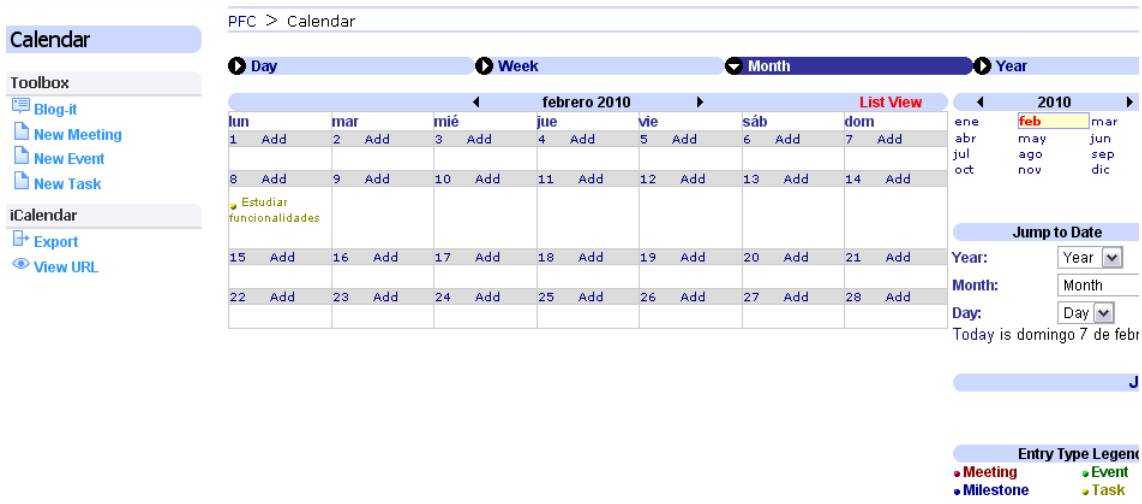


Ilustración 11: Visualización del calendario de un proyecto y posibles eventos

- Resources

Haciendo uso de Resources se podrá supervisar y organizar a cada miembro para la asignación más eficaz de tareas. Además en Resources, existirá un directorio completo de todos los participantes del proyecto.

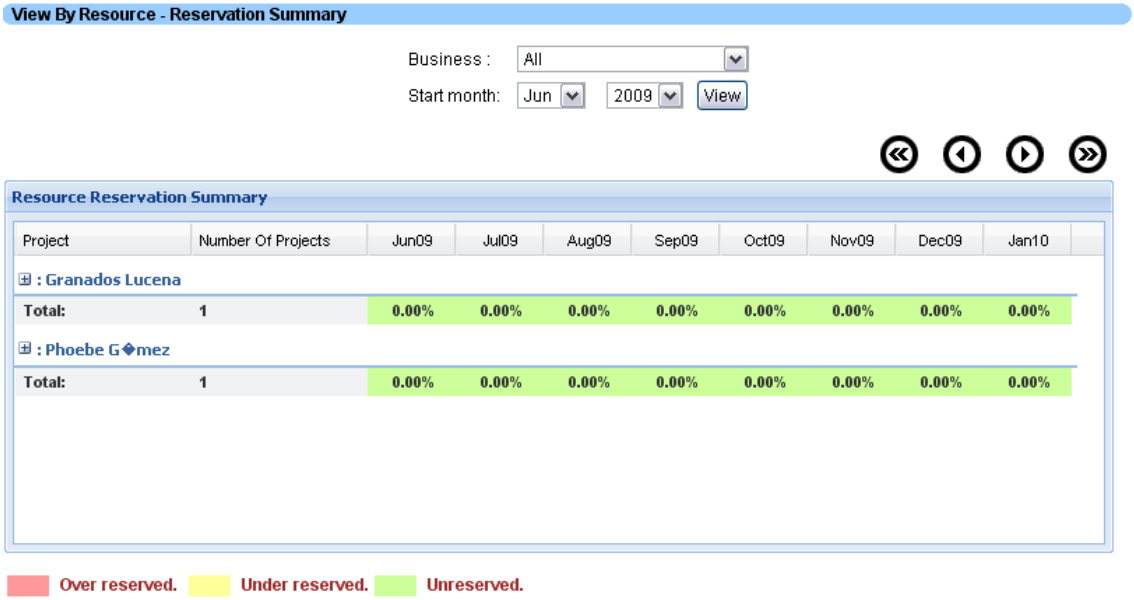


Ilustración 12: Visualización de la ocupación de los participantes

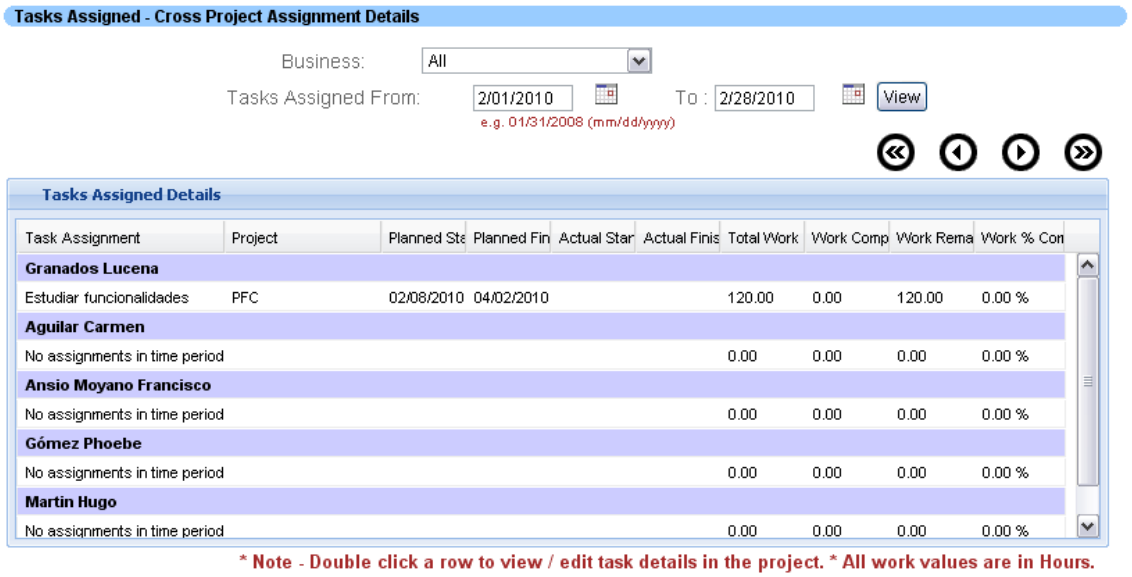


Ilustración 13: Visualización de la asignación de tareas a cada participante

- Business

Business supone otro espacio de trabajo diferente al Personal y al Projects. Puede verse como el departamento de una empresa, negocio, área, etc. Los proyectos que se crean dentro de Business pueden ser vistos sólo por aquellos miembros que pertenezcan al mismo también a ese business.

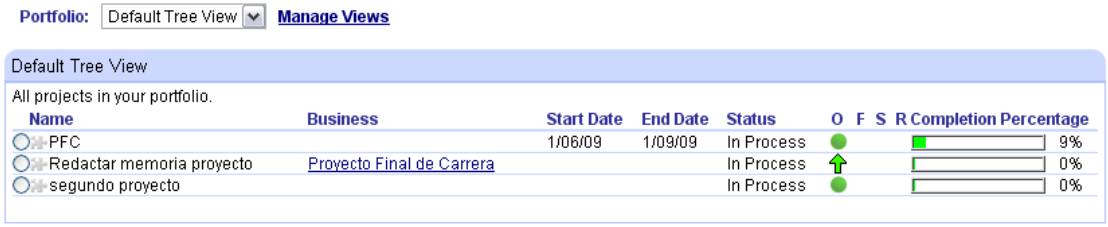


Ilustración 14: Listado de los proyectos en curso, y negocio al que pertenecen

Tras haber visto las diferentes funciones con las que se encuentra cada participante del proyecto en Project.net, se verá como se crea un proyecto nuevo y las propiedades de éste. La persona que lo crea será el jefe de proyecto, y sólo él, como ya se dijo anteriormente, podrá crear nuevas tareas dentro del proyecto.

Podremos definir las características del nuevo proyecto pinchando sobre New Project.

En primer lugar tendremos que introducir el nombre del proyecto. Si el proyecto pertenece a un departamento o "business" ya creado tendríamos que seleccionarlo, de igual forma, si fuese un subproyecto de otro. Tendremos que indicar también el área de la empresa al que pertenece y el tipo de moneda que se va a utilizar en los datos financieros.

El método para calcular el trabajo completado del proyecto puede hacerse de dos formas. Project.net puede calcular esa cifra automáticamente con el Workplan, o puede ser el propio jefe de proyecto el que la calcule, y vaya introduciendo ese valor manualmente.

Se seleccionará también la visibilidad del proyecto. Si es visible sólo para los participantes del proyecto, o para todos los que pertenezcan a un mismo departamento, o para cualquier miembro registrado.

El jefe del proyecto será el encargado de ir notificando el estado del proyecto. El estado de un proyecto puede estar "sin empezar", "en proceso", "bloqueado", "completado", o "en planificación". También puede notificarse un estado general que puede ser "mejorando", "sin cambios", "empeorando".

Posteriormente habrá que introducir posteriormente las fechas planeadas de inicio y fin del proyecto. Con esto ya se habrá completado los pasos a seguir para la creación de un proyecto.

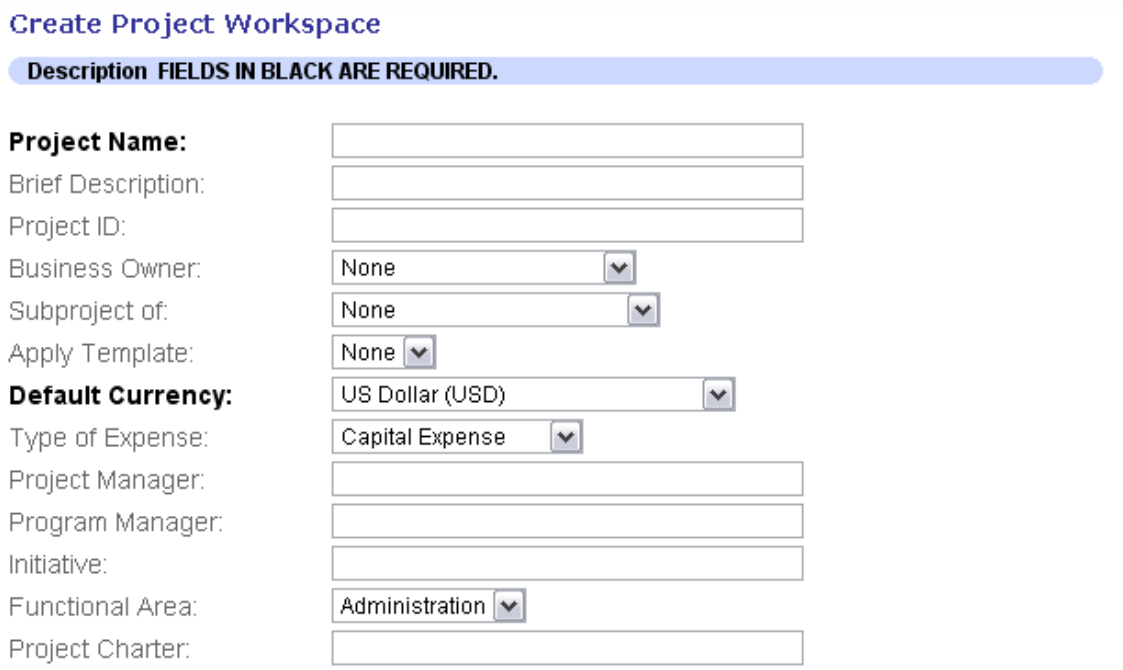


Ilustración 15: Características de un proyecto (I)

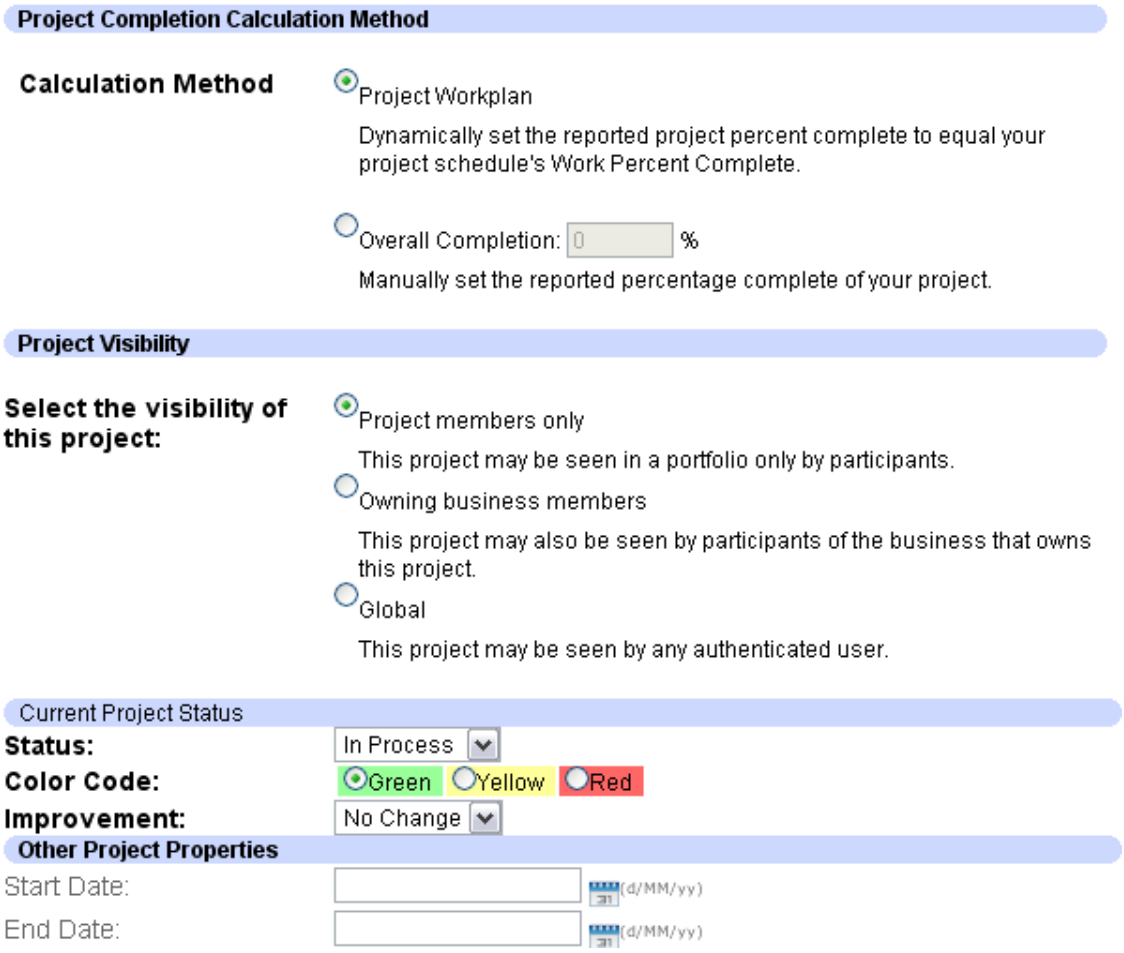


Ilustración 16: Características de un proyecto (II)

Una vez creado el proyecto, y a medida que las tareas y el tiempo vayan progresando, se puede ir informando al resto de participantes del estado en el que se encuentra el proyecto, desde varios puntos de vista, el general, el financiero, el temporal o el de recursos (participantes).

En la siguiente figura también se ve los datos financieros que puede introducir el jefe del proyecto sobre el mismo, que son los costes totales presupuestados, los estimados y los costes actuales, y el ROI estimado.

El ROI puede obtenerse como ROI= (B/Ci)*100 donde B y Ci son los beneficios y los costes iniciales, respectivamente. La forma aritmética de obtenerlo es ROI= (Vf/Vi)-1, donde Vf es la inversión final y Vi la inicial.

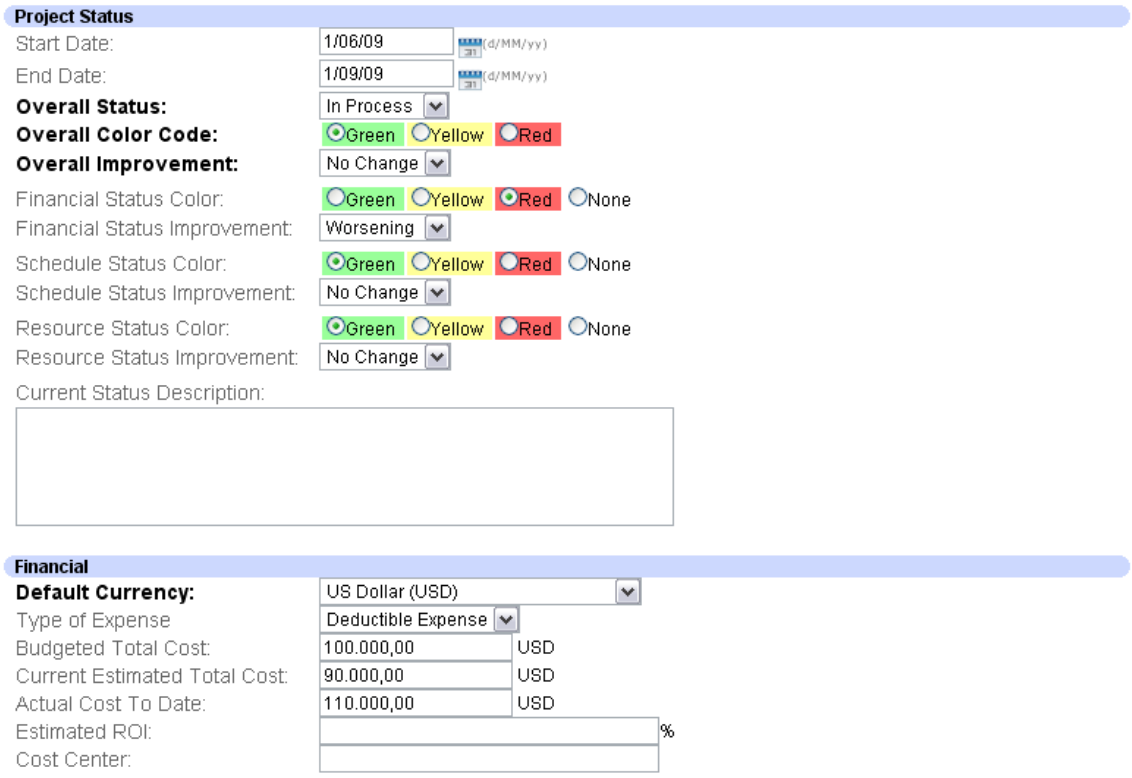


Ilustración 17: Características de un proyecto (III)

Toda esta información sobre el estado del proyecto es visible por el resto de participantes del mismo. En la figura siguiente se puede ver como Project.net calcula el coste excedente del presupuesto.

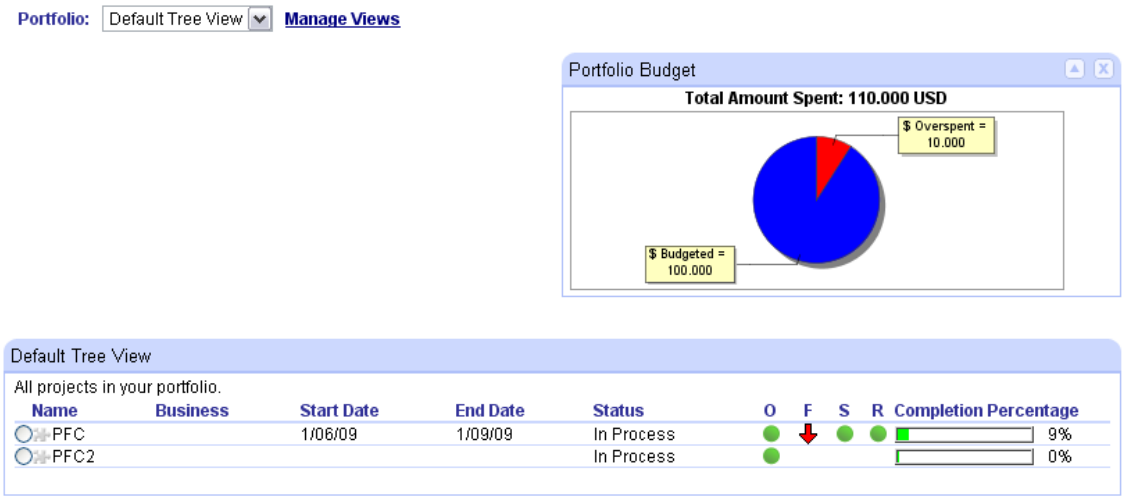


Ilustración 18: Visualización de los estados de los proyectos en curso

4.3. Experimentación

4.3.1. Instalación

En este apartado se explicará todo el proceso de implantación de Project.net en una máquina en Windows.

En primer lugar habrá que descargarse el fichero .zip de la siguiente web http://sourceforge.net/projects/projectnet. Una vez descargado, se extrae y se ejecuta el archivo starInstaller.bat. Automáticamente se cargará una página en el explorador que se tenga configurado por defecto.

Una vez hecho esto, habrá que seguir una serie de pasos que vienen muy bien explicados en la web http://dev.project.net/trac/pnet-community/wiki/installer_user_guide [4].

0) Aceptación licencia y registro.

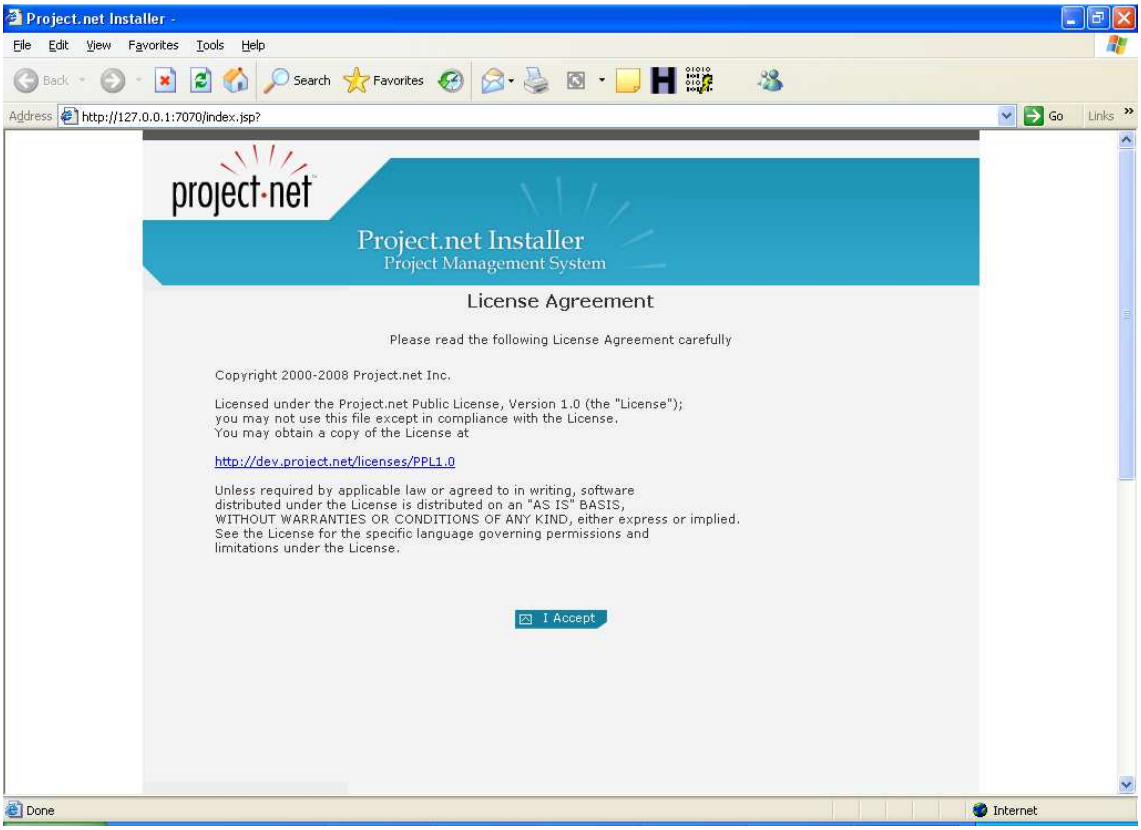


Ilustración 19: Instalación (I)

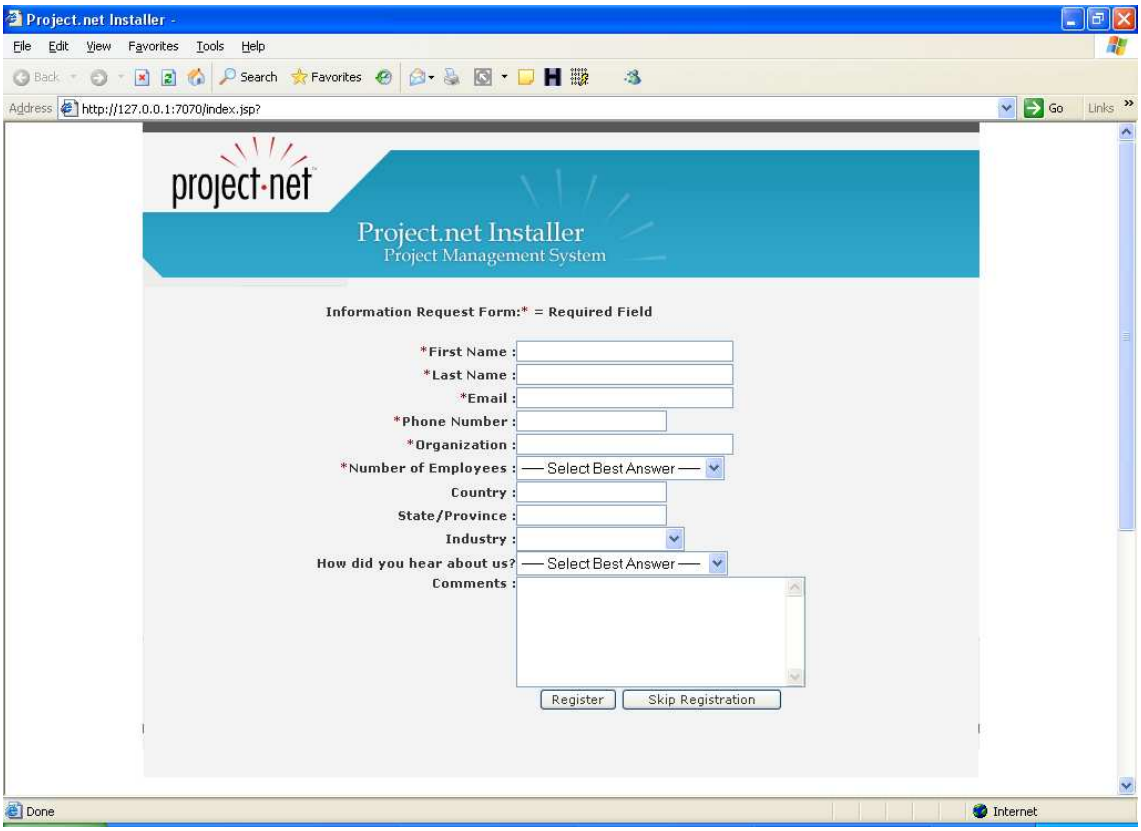


Ilustración 20: Instalación (II)

1) Configuración básica.

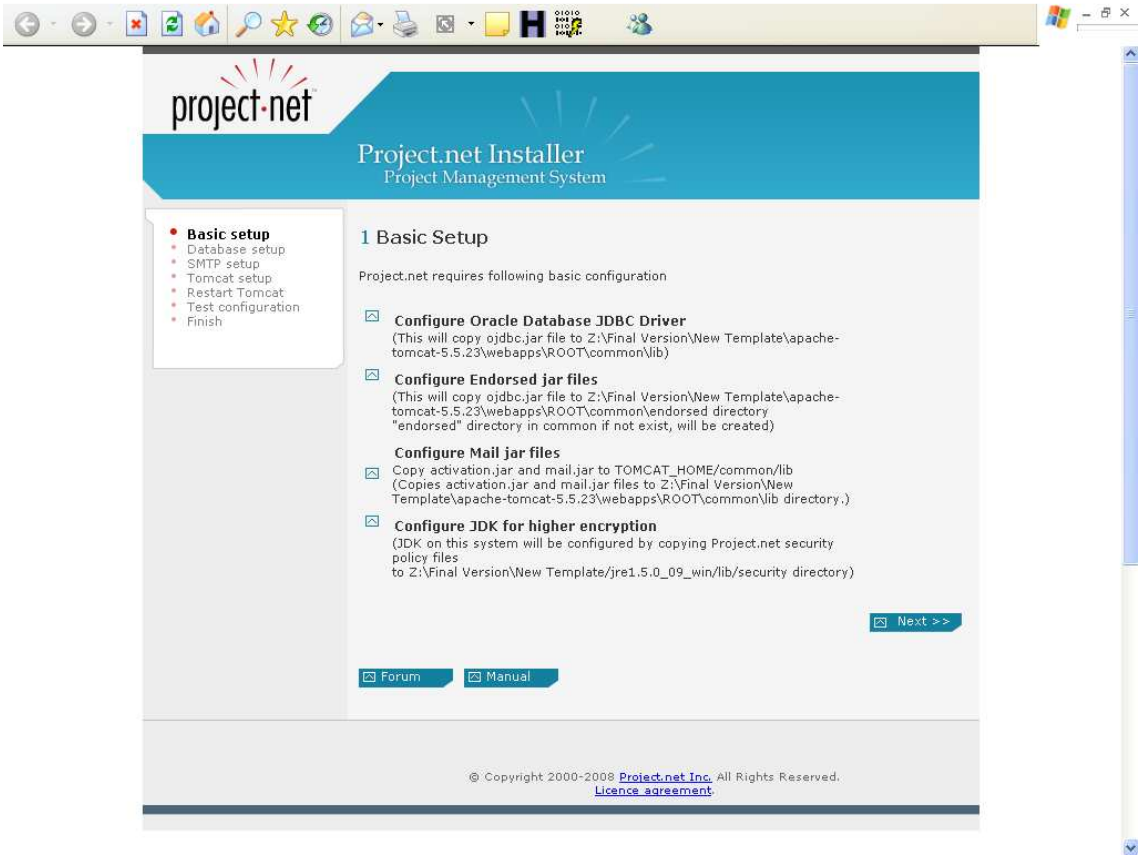


Ilustración 21: Instalación (III)

2) Configuración de las propiedades de la base de datos para ser usada por la aplicación. En primer lugar pregunta si existe ya una base de datos instalada, en caso contrario, no es posible seguir con la instalación de Project.net. Posteriormente, habrá que indicar el host y el nombre de la base de datos, el camino de la carpeta oradata, y el nombre de usuario y contraseña de la base de datos. La base de datos que se ha utilizado para llevar a cabo este proyecto es Oracle Database 10g Express Edition (Oracle Database XE), que es una base de datos de entrada, que puede desarrollarse, implementarse y distribuirse sin cargo ninguno. Se puede descargar fácilmente desde la web http://www.oracle.com/lang/es/database/Express_Edition.html . Esta base de datos puede instalarse en máquinas host de cualquier tamaño con cualquier cantidad de CPUs (una base de datos por máquina), no obstante XE almacenará hasta 4 GB de datos de usuario, utilizará hasta un 1 GB de memoria, y utilizará una sola CPU en la maquina host [11].

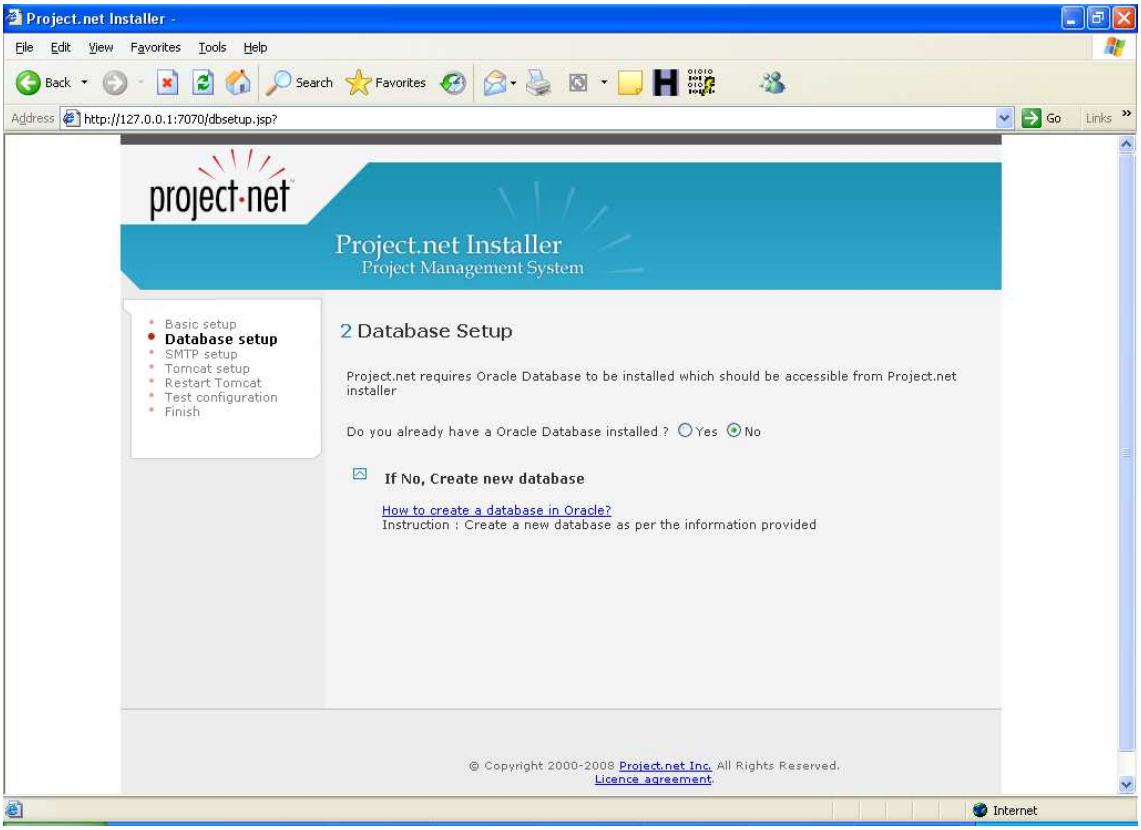


Ilustración 22: Instalación (IV)

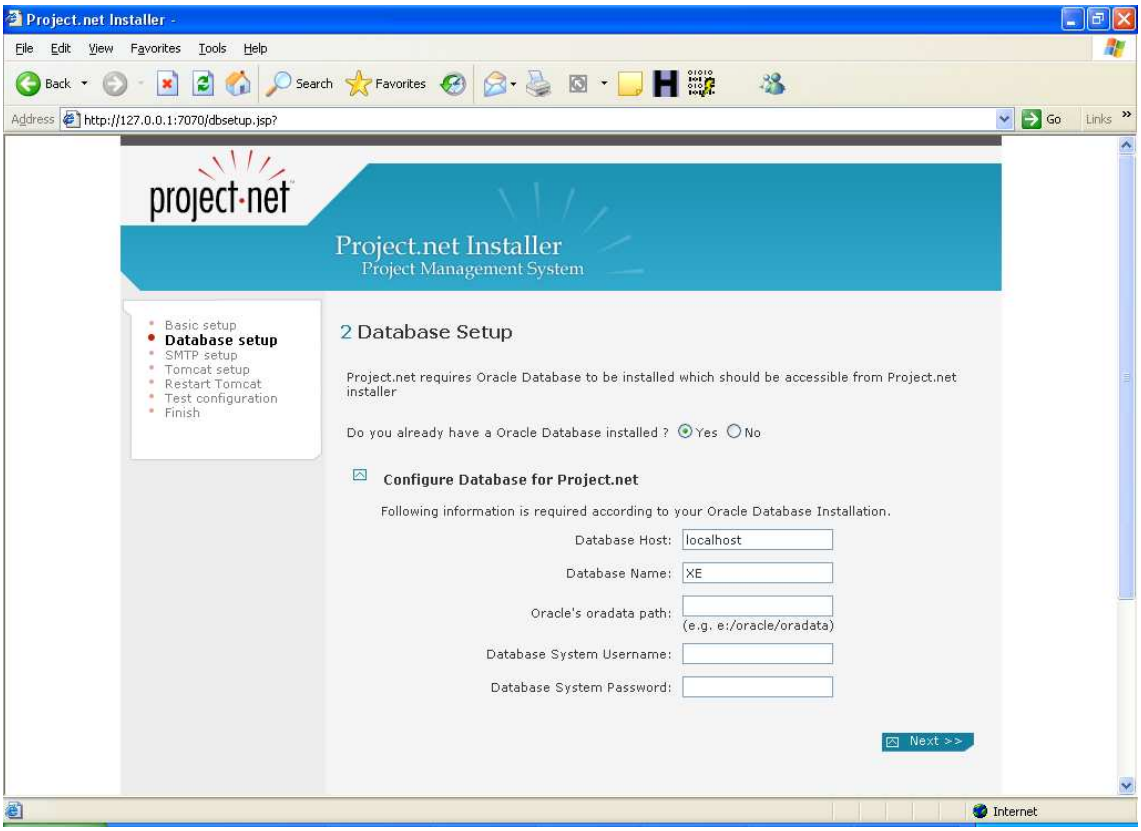


Ilustración 23: Instalación (V)

3) Configuración servidor de correo SMTP. En este paso, como primera opción, se instaló en la máquina física un servidor de correo SMTP, pero al tener contratada a la proveedora de servicios de internet una IP dinámica, la mayoría de correos enviados eran devueltos al provenir de un servidor con IP dinámica. Por lo que se optó, a usar el servidor SMTP de gmail. La configuración de conexión con el servidor de correo de gmail, se explicará en el apartado Recursos JNDI: Conectividad a base de datos Oracle y servidor de correo.

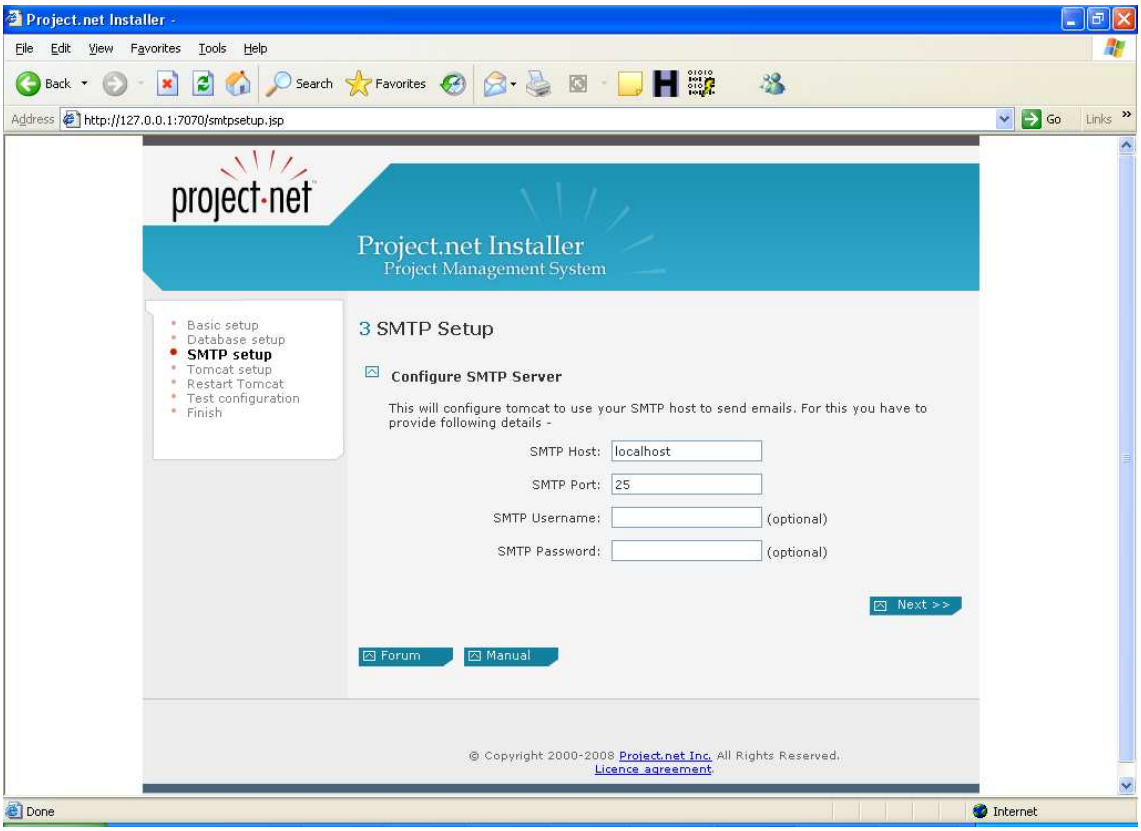


Ilustración 24: Instalación (VI)

4) Configuración de Tomcat. En este paso, una vez configurados la base de datos y el servidor SMTP, se cambiará el fichero context.xml con la información proporcionada en los dos pasos anteriores.

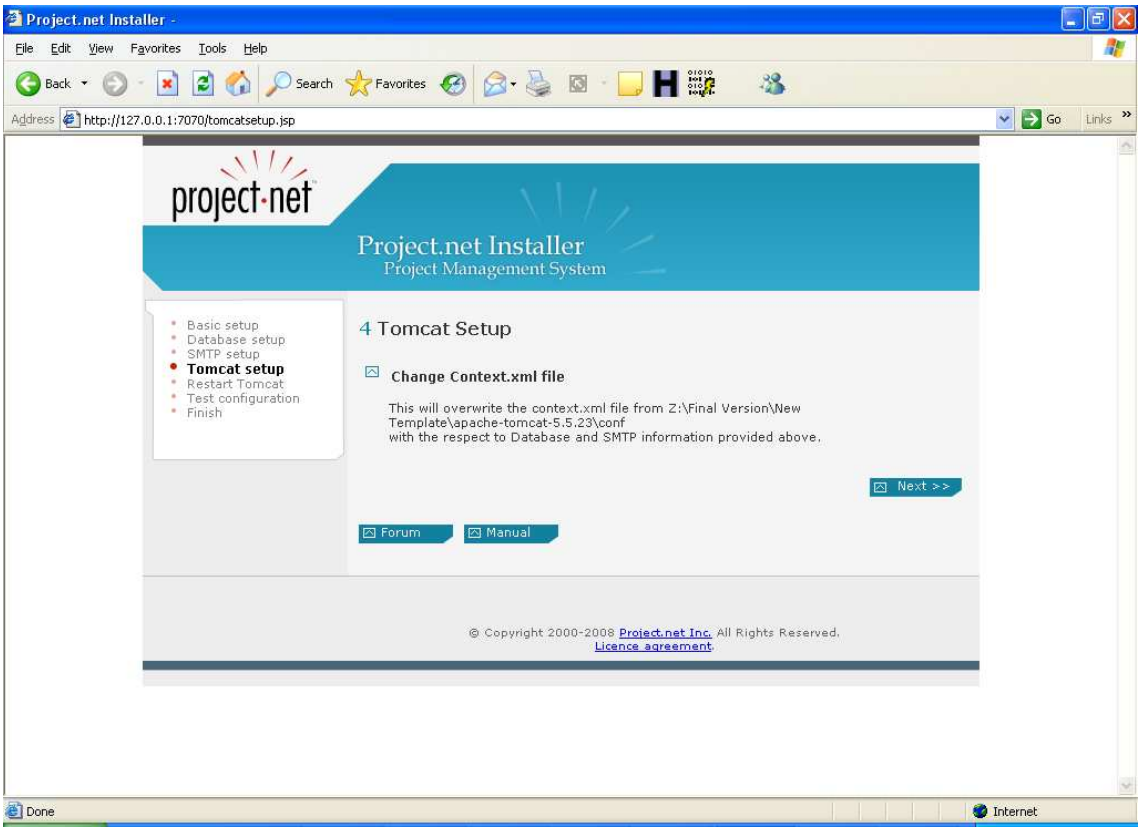


Ilustración 25: Instalación (VII)

5) Reinicio de Tomcat, puesto que se paró al finalizar la configuración del paso 4.

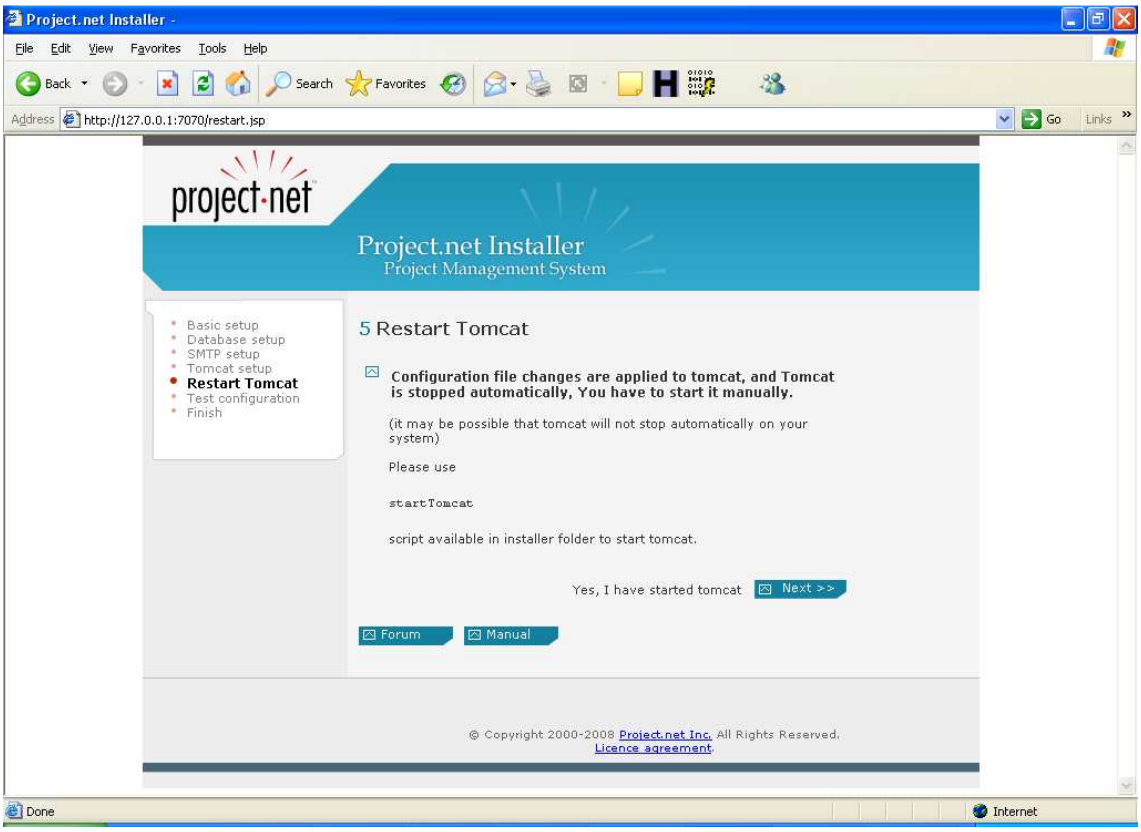


Ilustración 26: Instalación (VIII)

6) Test de configuración. Se chequea la configuración realizada en apartados anteriores.

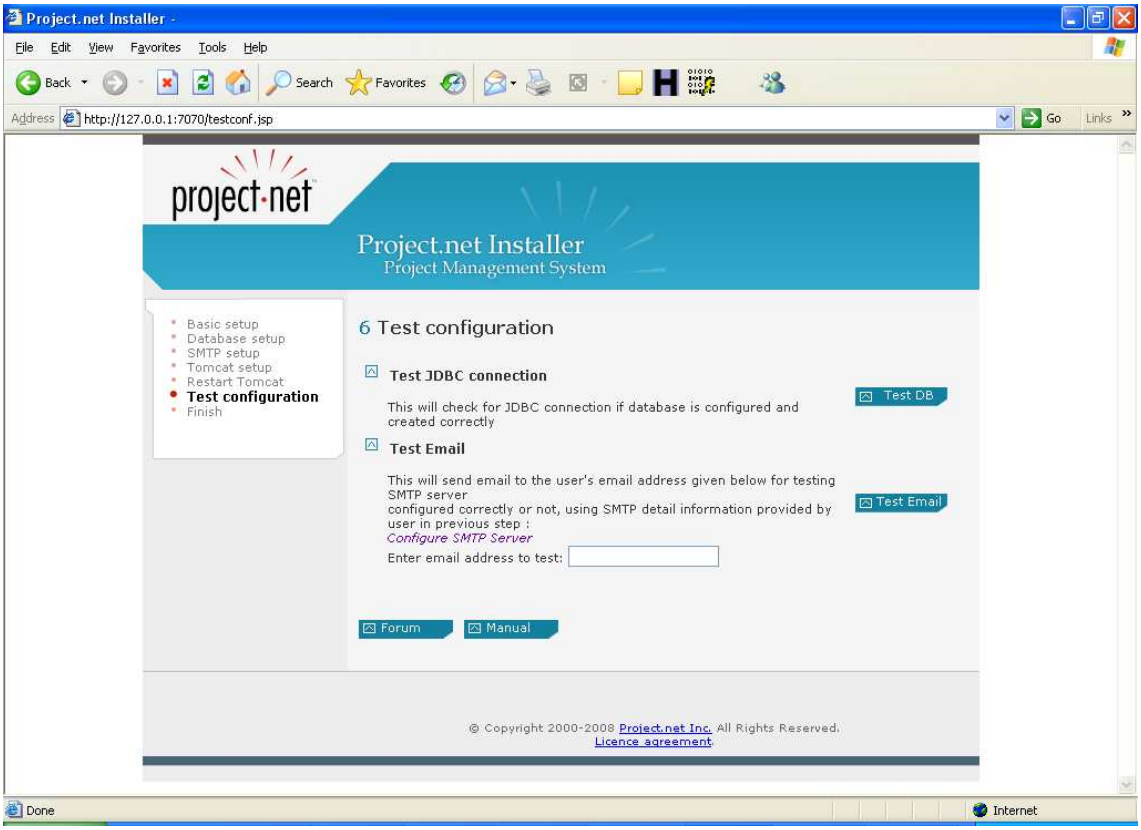


Ilustración 27: Instalación (IX)

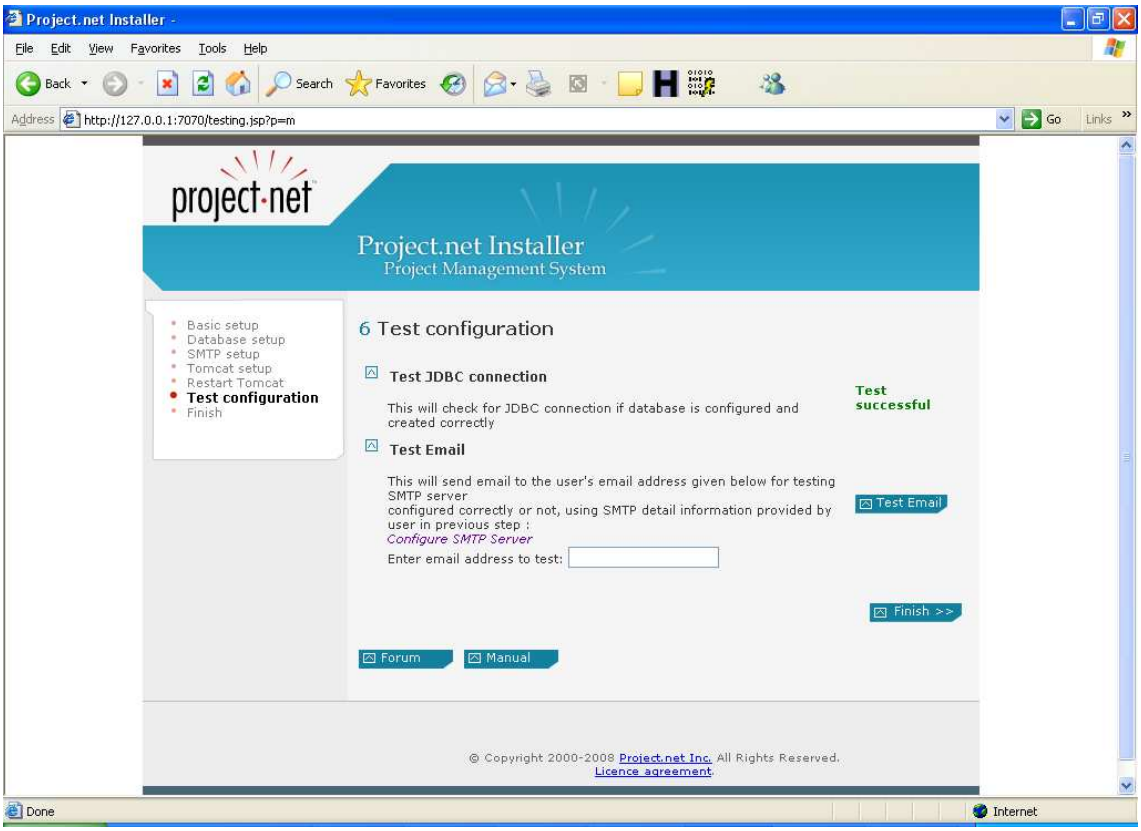


Ilustración 28: Instalación (X)

7) Reinicio de Tomcat, y login para entrar en la aplicación. Los nombre de usuario y contraseña por defecto son ambos "appadmin".

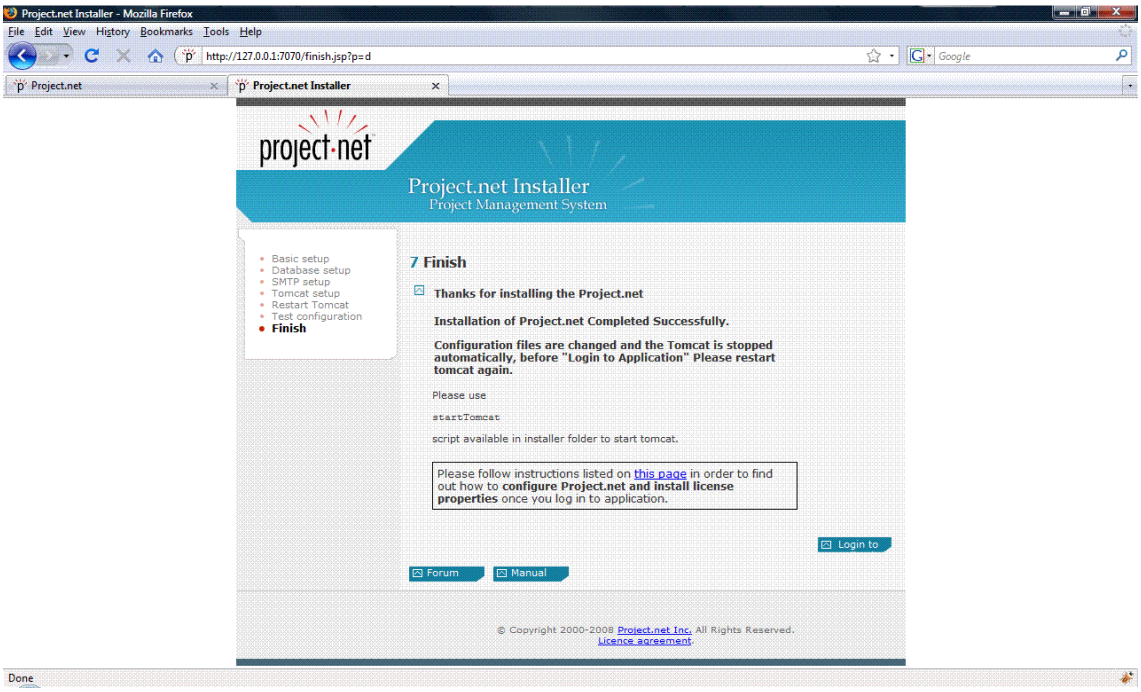


Ilustración 29: Instalación (XI)

8) Configuración de otras características de la herramienta. Se deberá actualizar la cuenta del administrador con otro nombre, configurar algunas

características de la aplicación, como son las carpetas donde se guardarán los ficheros que se suban a Project.net o la dirección externa que tendrá, entre otras propiedades, así como instalar una licencia, que se puede descargar de la siguiente dirección http://dev.project.net/trac/pnet-community/wiki/Downloads. Una vez hecho esto, ya se podrá crear una cuenta regular de usuario, y a partir de ella invitar al resto de participantes. El correo recibido en este último paso con el que se crea ya, una cuenta regular es el siguiente:

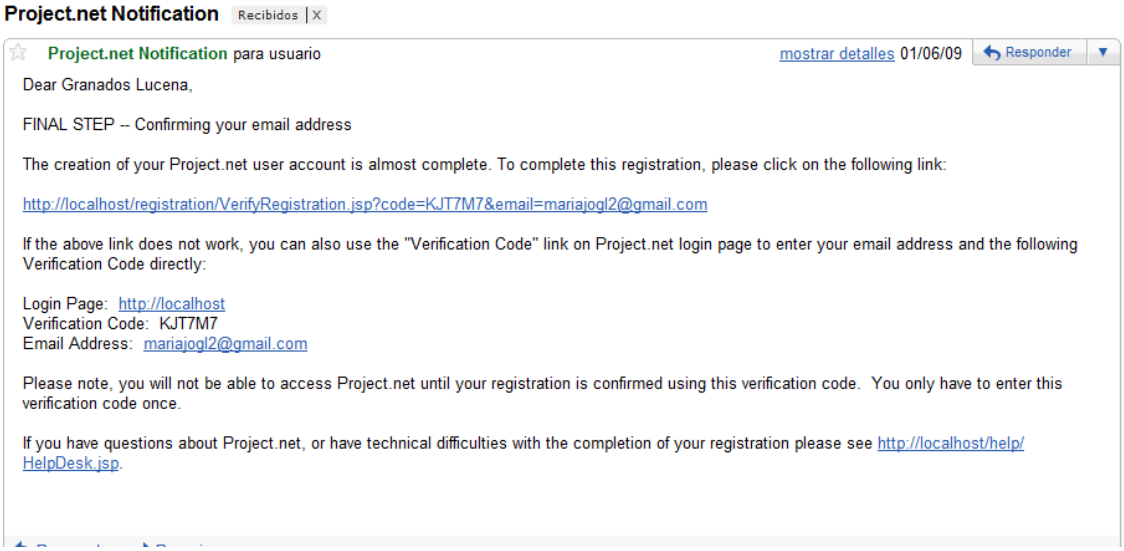
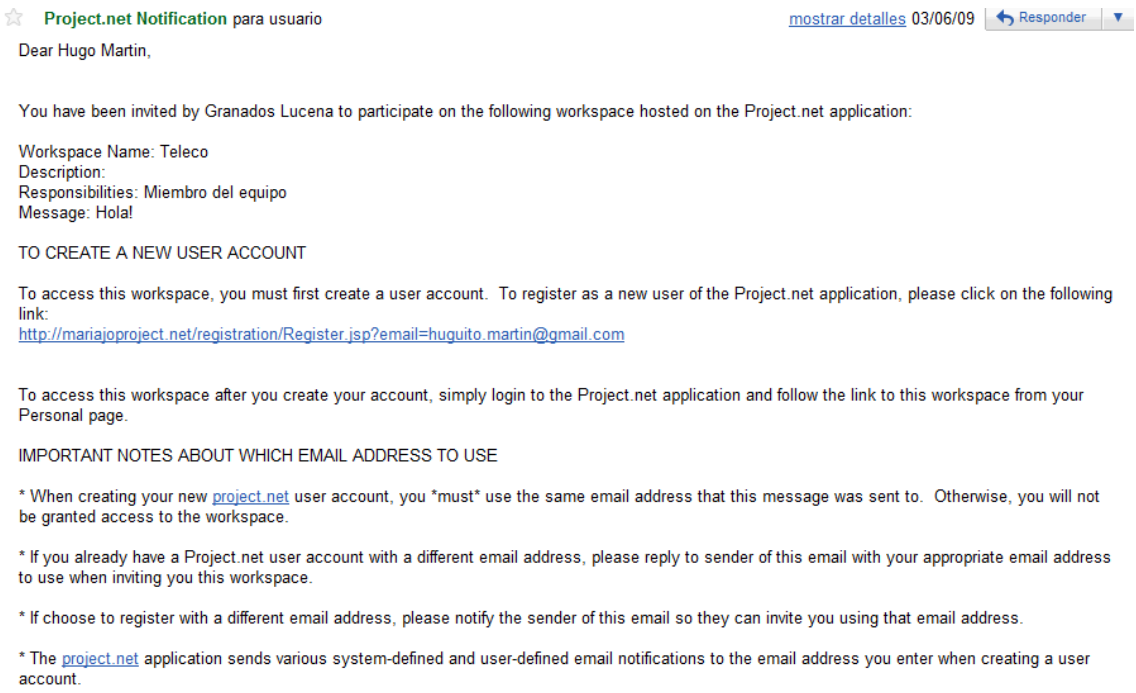


Ilustración 30: Correo que recibe el administrador

Con esa cuenta, podrá invitar a más participantes al proyecto o crear proyectos nuevos. El contenido del correo que recibirán los invitados será el siguiente:



Project.net Notification para usuario mostrar detalles 03/06/09 Responder

Dear Hugo Martin,

You have been invited by Granados Lucena to participate on the following workspace hosted on the Project.net application:

Workspace Name: Teleco
Description:
Responsibilities: Miembro del equipo
Message: Hola!

TO CREATE A NEW USER ACCOUNT

To access this workspace, you must first create a user account. To register as a new user of the Project.net application, please click on the following link:
http://mariajoproject.net/registration/Register.jsp?email=huguito.martin@gmail.com

To access this workspace after you create your account, simply login to the Project.net application and follow the link to this workspace from your Personal page.

IMPORTANT NOTES ABOUT WHICH EMAIL ADDRESS TO USE

* When creating your new project.net user account, you *must* use the same email address that this message was sent to. Otherwise, you will not be granted access to the workspace.

* If you already have a Project.net user account with a different email address, please reply to sender of this email with your appropriate email address to use when inviting you this workspace.

* If choose to register with a different email address, please notify the sender of this email so they can invite you using that email address.

* The project.net application sends various system-defined and user-defined email notifications to the email address you enter when creating a user account.

Ilustración 31: Correo que reciben el resto de participantes

Pinchando sobre el enlace, tendrán que registrarse y para completar el registro deberán introducir un código que habrán recibido anteriormente en su correo.
Y por último, para terminar este apartado, anteriormente, se ha dicho que se puede configurar el nombre del sitio, es decir, la dirección que tendrán que escribir los participantes para poder entrar en la aplicación. Al tratarse, de una IP dinámica, y si se quería que Project.net fuese accesible desde cualquier parte del mundo, ese nombre del sitio debería estar continuamente relacionado con la IP que tuviese el servidor en cada momento. Para conseguir esto, se hizo uso de DynDNS y un programa que, instalado en la máquina, actualiza automáticamente la IP e informa a DynDNS, previa configuración en el router.

4.3.2. Configuración

4.3.2.1. Configuración básica de Tomcat

En el directorio, /conf de Tomcat 5 se encuentran diversos archivos de configuración. Se irá explicando en detalle cada uno de ellos [3].

- Server.xml: El archivo de configuración principal y que el servidor lee al iniciarse.

Se muestra a continuación nuestro fichero server.xml, suprimiendo comentarios.

```
-<Server port="8005" shutdown="SHUTDOWN">

…

-<Service name="Catalina">


        <Connector port="7070" maxHttpHeaderSize="8192"
maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
enableLookups="false" redirectPort="8443" acceptCount="100"
connectionTimeout="20000" disableUploadTimeout="true" />

        <Connector port="8009" enableLookups="false" redirectPort="8443"
protocol="AJP/1.3" />

-<Engine name="Catalina" defaultHost="localhost">

        <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
resourceName="UserDatabase" />

        <Host name="localhost" appBase="webapps" unpackWARs="true"
autoDeploy="true" xmlValidation="false" xmlNamespaceAware="false">

      /Host>
</Engine>
</Service>
</Server>
```

En este fichero se ve que se configura un único servicio dentro de una única instancia del componente servidor que se corresponde con al elemento <server> de XML. La primera línea activa define el componente servidor. Se indica a Tomcat que inicie una instancia de servidor (una JVM), que escuche en el puerto 8005 un comando de cierre. El comando de cierre contendrá el texto SHUTDOWN, lo que permite al administrador o al software de la consola de administración cerrar este servidor Tomcat. La instancia del servidor no imprimirá mensajes de depuración ya que la depuración se ha establecido en 0.

Dentro del elemento <server> se permiten los subelementos XML siguientes:

- <service>: Grupo de conectores asociados a un motor. Los conectores controlan diferentes protocolos cliente (HTTP, HTTPS, JK2 y otros) y gestionan la concurrencia de solicitudes, mientras que el motor las procesa.
- <Listener>: Escuchador para interceptar los eventos del servidor (start, stop, before start, before stop, after star, after stop).
- <GlobalNamingResources>: Recursos JNDI definidos para estar globalmente disponibles en esta instancia del componente.

La siguiente línea del archivo se define un componente de servicio. El principal objetivo de este componente es agrupar un motor de procesamiento de solicitudes con su conjunto configurado de conectores de control de protocolos. Se trata de un componente de nivel superior. En este caso, se ha definido una instancia de servicio con el nombre Catalina. Los subelementos que puede tener el elemento <service> son Connector y Engine.

Antes de explicar los componentes conectores tenemos que tener en cuenta que nuestro Tomcat funciona en modo independiente, es decir, todas las páginas estáticas y archivos de gráficos de Project.net se sirven directamente desde el servidor Tomcat. En este modo, no se necesita un servidor Web adicional ya que Tomcat actúa tanto como servidor Web como contenedor JSP/Servlet. Tomcat utiliza su conector HTTP para procesar la solicitud HTTP entrante, con lo que desaparece la necesidad de un servidor Web externo.

En nuestro fichero server.xml se define un conector HTTP 1.1 para el servicio Catalina, y un conector JK2 por si fuera necesario [15].

A continuación, se define el único componente motor asociado al servicio Catalina. Un motor es un contenedor y, básicamente, representa una instancia del procesador Servlet. A esta instancia de motor configurado se le ha asignado el nombre de Catalina. Un motor puede procesar una solicitud destinada a varios host virtuales. El atributo defaultHost especifica el host virtual al que Tomcat dirigirá la respuesta si ésta no se destina específicamente a uno de los host virtuales configurados en el archivo server.xml. debug=”0” indica que no se escribirán mensajes de depuración específicos del motor en el registro, sólo los mensajes de errores muy graves.

Como contenedor, el elemento <Engine> puede contar con elementos adicionales:

- Host: Cada elemento <Host> especifica un host virtual controlado por el motor. Tomcat 5 puede controlar varios host virtuales por instancia de motor o servicio. Es similar a una de las características más conocidas del servidor Web Apache.
- DefaultContext: Crea un contexto (colección de parámetros para propiedades y elementos configurables) para las aplicaciones Web que se implementan automáticamente al iniciar Tomcat. Las propiedades especificadas en este contexto predeterminado también están disponibles para todas las aplicaciones Web que se ejecuten dentro del motor.
- Logger: Especifica la instancia de componente de registro utilizada por el motor para registrar mensajes. A menos que se reemplace por medio de contenedores internos, es la instancia de registrador predeterminada para todos los componentes anidados del motor.

- Realm: Este reino se utiliza de forma predeterminada para compatibilidad de seguridad declarativa para asignar funciones a usuarios y para funciones de autenticación. Los elementos <Host> y <Context> de cada host virtual pueden tener su propio reino para ello. En caso contrario, se utiliza el reino configurado en el nivel del motor.
- Valve: Las válvulas añaden lógica de procesamiento a la secuencia de control de solicitudes y respuestas en el nivel del motor.
- Listener: Se utiliza para configurar escuchadores que controlan el inicio y detección del motor.

En nuestro caso, el primer componente anidado dentro del motor es el componente Reino. Se configura un reino UserDatabase para cargar el archivo tomcat-users.xml en memoria para utilizarlo en labores de autenticación en aplicaciones predeterminadas como admin y manager.

Un reino es un mecanismo de seguridad utilizado para realizar la autenticación e implementar seguridad controlada por contenedores [15]. Básicamente, los reinos son orígenes de datos que asignan nombres de usuario y contraseña (para la autenticación), y nombres de usuario y funciones (para seguridad). Por ejemplo, en nuestro caso, hemos creado un nombre de usuario 'carmen' con una contraseña 170883.

El siguiente componente anidado es el componente Host. Se trata de un contenedor que puede incluir otros componentes anidados. Representa un host virtual controlado por la instancia del servidor Tomcat. Se configura como elemento <Host> dentro del archivo server.xml. Todos los elementos <Host> definidos dentro del elemento <Engine> representan a otro host virtual controlado por dicho motor.

En nuestro caso, se define un host virtual con el nombre localhost, que coincide con el defaultHost especificado en el contenedor <Engine>. Las aplicaciones, en este caso, sólo una, Project.net, que se implementen para este host virtual se encuentran en el directorio /webapps. El atributo unpackWARs indica que si Tomcat encuentra cualquier archivo WAR en el directorio appBase, se descomprimirán antes de que se ejecute la aplicación Web. Si establece unpackWARs en false, Tomcat ejecuta Project.net sin descomprimirla, por lo que se ahorra espacio pero se reduce el rendimiento. El atributo autoDeploy se establece en true, de forma que Tomcat analiza la inclusión de nuevas aplicaciones Web o los cambios realizados en la ya existentes y, tras ello, las implementa automáticamente o las vuelve a implementar.

- Server-minimal.xml: Es un archivo server.xml reducido, similar al anterior pero sin los comentarios ampliados ni compatibilidad con funciones opcionales.

- Tomcat-users.xml: Archivo que contiene información sobre autenticación de usuarios y funciones para configurar un reino de base de datos de usuarios. De forma predeterminada, las aplicaciones admin y manager de Tomcat lo utilizan. Sólo los usuarios con la función admin podrán acceder al configurador Web y sólo los usuarios con la función manager podrán acceder a la aplicación manager. Este tipo de aplicaciones serán totalmente invisibles para los usuarios de Project.net, tanto para los usuarios invitados como para el administrador.

Nuestro archivo tomcat-users.xml es el que se muestra a continuación.

```
?xml version="1.0" encoding="utf-8" ?>
<tomcat-users>
    < role rolename="tomcat" />
    <role rolename="role1" />
    <role rolename="manager" />
    <role rolename="admin" />
    <user username="tomcat" password="tomcat" roles="tomcat" />
    < user username="role1" password="tomcat" roles="role1" />
    <user username="both" password="tomcat" roles="tomcat,role1" />
     <user username="manager" password="manager" roles="admin,manager"
/>
     <user username="admin" password="" roles="admin,manager" />
</tomcat-users>
```

- web.xml: Archivo descriptor de implementación predeterminado para todas las aplicaciones Web que se implementen en esta instancia del servidor Web. Proporciona definición básica de Servlet y asignaciones MIME comunes a todas las aplicaciones Web. También actúa como descriptor de implementación de todas las aplicaciones Web que no cuenten con uno propio.

Según la especificación Servlet 2.4, todas las aplicaciones Web deben incluir un descriptor de implementación (el archivo web.xml), que debe alojarse en el directorio WEB-INF/ de la aplicación Web.

También existe un archivo web.file en el directorio /conf, similar al archivo web.xml de la aplicación. Sin embargo, éste se utiliza para especificar las propiedades predeterminadas de todas las aplicaciones Web que se ejecuten en esta instancia

del servidor. A continuación veremos qué propiedades predeterminadas se configuran en este archivo web.xml. El archivo es el siguiente.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd" version="2.4">

<servlet>
    <servlet-name>default</servlet-name>
    <servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-
class>
 <init-param>
    <param-name>debug</param-name>
    <param-value>0</param-value>
    </init-param>
 <init-param>
    <param-name>listings</param-name>
    <param-value>false</param-value>
 </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet>
    <servlet-name>jsp</servlet-name>
    <servlet-class>org.apache.jasper.servlet.JspServlet</servlet-class>
 < init-param>
    <param-name>fork</param-name>
    <param-value>false</param-value>
</init-param>
 <init-param>
    <param-name>xpoweredBy</param-name>
    <param-value>false</param-value>
</init-param>
    <load-on-startup>3</load-on-startup>
</servlet>

<servlet-mapping>
    <servlet-name>default</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>

 <servlet-mapping>
    <servlet-name>jsp</servlet-name>
      url-pattern>*.jsp</url-pattern>
</servlet-mapping>
```

```
   <servlet-mapping>
   <servlet-name>jsp</servlet-name>
   <url-pattern>*.jspx</url-pattern>
</servlet-mapping>

….
<mime-mapping>
   <extension>jpeg</extension>
   <mime-type>image/jpeg</mime-type>
</mime-mapping>
 <mime-mapping>
   <extension>jpg</extension>
   <mime-type>image/jpeg</mime-type>
</mime-mapping>
<mime-mapping>
   <extension>js</extension>
   <mime-type>text/javascript</mime-type>
</mime-mapping>
 ….

<welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
</web-app>
```

En primer lugar, vemos un encabezado XML estándar y una referencia al esquema Servlet 2.4.

En las siguientes líneas, se define un <servlet>, donde se especifica un Servlet predeterminado que se utiliza para servir recursos estáticos (archivos HTML estáticos, archivos GIF, etc.) a Project.net.

A continuación, se define JspServlet para que convierta páginas JSP en Servlet y los ejecute.

Un elemento <servlet-mapping> especifica cómo se procesan las solicitudes entrantes que contengan un patrón de URL concreto. Se especifica lo siguiente: "Al ver un URL que coincida con el patrón /, se redirige al Servlet predeterminado".

Los dos siguientes elementos <servlet-mapping> especifican que los URL que incluyan *.jsp y *.jspx deben pasarse al Servlet jsp para su procesamiento. En el elemento <server-mapping> anterior, el Servlet jsp es la clase org.apache.jasper.servlet.JspServlet.

El siguiente conjunto de elementos <mime-mapping> predeterminados. Tomcat 5 utiliza estas asignaciones para servir al cliente archivos estáticos con extensiones específicas. Al transmitir el archivo al cliente (habitualmente a un navegador), genera un encabezado HTTP Content-Type. La mayoría de los navegadores utilizan una aplicación de ayuda para procesar el archivo transmitido si reconocen el tipo de contenido especificado. Por ejemplo, Microsoft Internet Explorer puede iniciar el programa Media Player cuando detecta un tipo de contenido video/x-mpeg. Se trata solamente de las asignaciones predeterminadas; el propio descriptor de implementación de Project.net (el archivo web.xml) puede reemplazar o ampliar esta lista. (En la muestra del fichero web.xml no he copiado todas las asignaciones mime, hay muchas más).
La última sección del archivo web.xml solo se aplica a Tomcat en modo independiente como es nuestro caso. El servlet predeterminado examinará el directorio raíz del host virtual y buscará index.html, index.htm o index.jsp para representarlo. Project.net puede reemplazar esta lista en su propio descriptor de implementación.

- catalina.policy: Java 2 cuenta con un refinado modelo de seguridad que permite al administrador controlar la accesibilidad de los recursos del sistema. Aunque de forma predeterminada, Tomcat 5 se inicia en modo no seguro, este archivo es muy importante en instalaciones seguras del servidor.

- Catalina.properties: Tomcat 5 lee y utiliza las propiedades de este archivo durante el inicio. Proporciona acceso de paquetes internos y control de definición, así como control sobre los contenidos de los cargadores de clases de Tomcat, como veremos en otro apartado de esta memoria.

4.3.2.2. Configuración Aplicación Web: Project.net

Las aplicaciones Web están formadas por contenido estático (como páginas HTML y archivos de imagen) y contenido dinámico (como Servlet, JSP y clases de Java).
En este apartado analizaremos los aspectos relacionados con la configuración de Project.net: estructura y contenido de la aplicación, y el descriptor de implementación de Project.net.

- Contenidos de una aplicación Web

Las aplicaciones Web se suelen instalar en el directorio /webapps. La aplicación Web se implementa en un directorio que suele tener el mismo nombre que la

aplicación y que se utiliza en el URL de ésta. Por ejemplo, si una aplicación se encuentra en el directorio exampleapp, al que se puede acceder desde el URL http://localhost:8080/exampleapp/. En este caso, /exampleapp/ es la ruta de contexto de la aplicación Web. Es la parte del URL que aparece por detrás del servidor y el número de puerto, y se utiliza para resolver la ubicación del recurso.
Una excepción es la aplicación ROOT, que está disponible cuando no se especifica una ruta de contexto. Es ésta la que utilizaremos nosotros. Para acceder a Project.net simplemente http://mariajoproject.dyndns.org:7070/
En lo que respecta a la estructura de la aplicación Web, como mínimo se necesita un directorio WEB-INF con un archivo web.xml en su interior. Como ya vimos anteriormente. Todos los contenidos de los directorios WEB-INF y META-INF pertenecen a la categoría de recursos de una aplicación y no se puede acceder directamente a los mismos desde aplicaciones cliente. Todo lo que no esté en esos directorios son recursos públicos, a los que se accede por medio del correspondiente URL.
En la mayoría de los casos, al solicitar un recurso Web desde un navegador (como una página HTML), éste se sirve sin modificación por parte del servidor Web. Las páginas JSP son una excepción. Una página JSP se pasa primero a través de un compilador JSP que compila el archivo en un archivo Java y tras ello, compila el archivo Java en una clase Servlet. Seguidamente se ejecuta esta clase y el resultado se muestra en el navegador.
El código responsable de esta operación es una asignación URL definida por medio de un elemento <servlet-mapping>. Esta asignación URL se define en /conf/web.xml. Este archivo es el descriptor de implementación de todas las aplicaciones Web.

```
<servlet-mapping>
   <servlet-name>jsp</servlet-name>
     url-pattern>*.jsp</url-pattern>
</servlet-mapping>
```

El código anterior especifica que todos los URL que terminen en .jsp deben pasarse al Servlet jsp definido en el mismo archivo de configuración /conf/web.xml. La definición de este servlet es la siguiente:

```
<servlet>
    <servlet-name>jsp</servlet-name>
    <servlet-class>org.apache.jasper.servlet.JspServlet</servlet-class>
< init-param>
```

```
    <param-name>fork</param-name>
    <param-value>false</param-value>
</init-param>
<init-param>
    <param-name>xpoweredBy</param-name>
    <param-value>false</param-value>
</init-param>
    <load-on-startup>3</load-on-startup>
</servlet>
```

Como se puede apreciar, el nombre completo del Servlet es org.apache.jasper.servlet.JspServlet. El Servlet recibe la solicitud, utiliza la ruta de contexto para cargar la página JSP y la pasa al compilador JSP de Tomcat, denominado Jasper. Las opciones restantes establecen el nivel de registro de la compilación JSP y el proceso de ejecución, y garantizan que la clase de Servlet se carga en memoria al inicio con prioridad 3 (siendo 1 el valor superior) para asegurarse de que se carga antes de solicitar una página JSP.

Dentro del directorio WEB-INF, se encuentran los directorios clases y lib, a parte del descriptor de implementación.
El directorio classes contiene clases de Servlet y utilidades, incluyendo JavaBeans. También puede contener archivos de recursos como listas de mensajes clave/valor, con mensajes de error para la aplicación, e información de configuración específica de la aplicación.
El directorio lib contiene paquetes de bibliotecas de Java que la aplicación necesita y que se incluyen dentro de la misma. Los archivos JAR incluidos sólo están disponibles para la aplicación.

- El descriptor de implementación (web.xml)

Un descriptor de implementación es un archivo XML que contiene información de configuración que la aplicación Web utiliza para ejecutarse en el motor Servlet. Es similar al explicado ya en el apartado anterior [3]. En el anexo se entra en más detalle en este fichero de configuración [15].

4.3.2.3. Recursos JNDI: Conectividad a base de datos Oracle y servidor de correo

Dentro del archivo server.xml se pueden definir recursos JNDI (Interfaz de nombres y directorios de Java) y se pueden acceder a los mismos de forma compatible con J2EE desde Project.net. En nuestro caso se define un recurso que añade una implementación UserDatabase (para almacenar información de autenticación y funciones en Tomcat 5). El reino UserDatabase es un component de Tomcat 5 utilizado para implementar una base de datos de usuario, contraseña y funciones para labores de autenticación y seguridad gestionada por contenedores [3].

```
<Resource name="UserDatabase" auth="Container"
type="org.apache.catalina.UserDatabase" description="User database that
can be updated and saved"
factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
pathname="conf/tomcat-users.xml" />
```

JNDI es un API utilizado para buscar información relativa a la red. Se ha diseñado para su uso con cualquier servicio de nombres y directorios compatible, independientemente de su interfaz nativa. Los archivos de inclusión JNDI ofrecen información como la que indicamos a continuación:

- Nombre de usuario y contraseña (autenticación)
- Directiva de control de acceso (quién puede acceder a qué)
- Directorios de organización
- Servidores (correo electrónico, base de datos, etc.)

En la siguiente figura se ilustra cómo JNDI unifica el acceso a servicios de directorios entre diferentes redes.

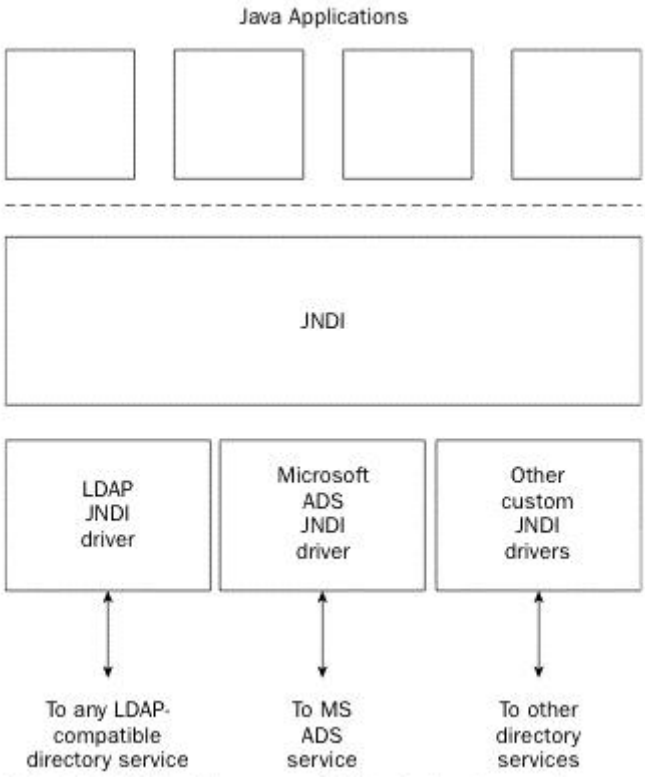


Ilustración 32: Arquitectura de los recursos JNDI, extraída de [3]

JNDI es la capa que proporciona una interfaz de programación uniforme a las aplicaciones, al tiempo que traduce los comandos de API en las operaciones específicas de la red que se envían a través de los controladores de sus complementos. De hecho, muchos de los sistemas de directorios modernos admiten el protocolo LDAP. JNDI obtiene compatibilidad con servicios nuevos o heredados a través de su controlador LDAP.

El papel del servidor Tomcat, en lo que respecta a su compatibilidad con JNDI, es muy interesante. De hecho, sólo se encarga de proporcionar funciones de búsqueda para cualquier aplicación Web. Tomcat es un servidor compatible con J2EE y Servlet 2.4 que facilita el uso de JNDI por parte de aplicaciones Web alojadas.

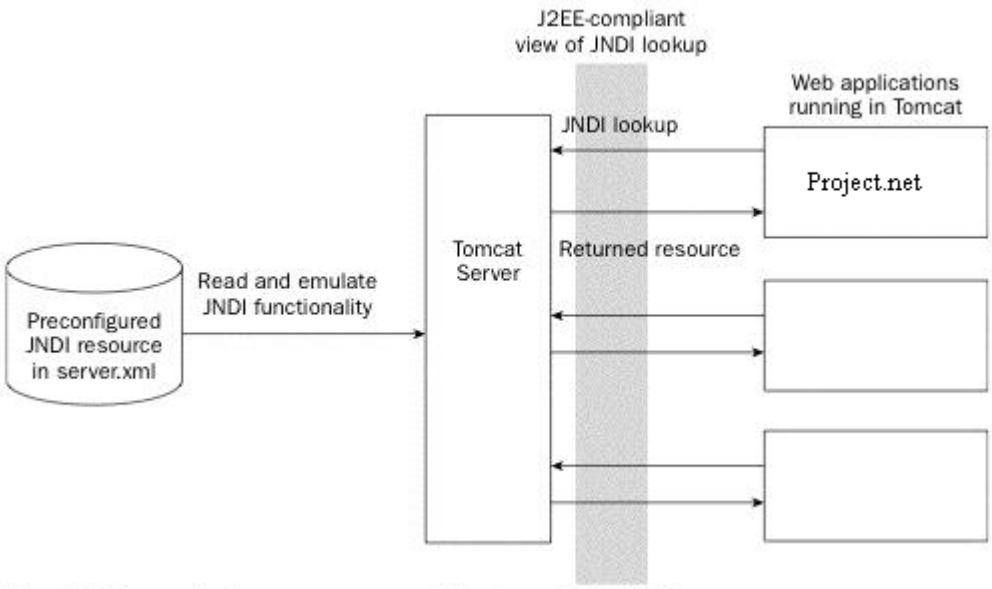


Ilustración 33: Interacción de Project.net, recurso JNDI y Apache Tomcat, extraída de [3]

En la imagen, la aplicación Web, Project.net, que se ejecuta en Tomcat recupera determinados recursos JNDI a través de una convención de programación estándar y API. Esto permite colocar las solicitudes de forma independiente al servidor de aplicaciones, lo que permite que las aplicaciones se puedan utilizar entre servidores de diferentes distribuidores.

El contenedor de Tomcat intercepta las solicitudes JNDI estándar de la aplicación. Para completar estas solicitudes de API JNDI, Tomcat debe comprobar su conjunto de recursos predefinidos (en el archivo server.xml) para determinar qué necesita devolver a la aplicación. Básicamente, Tomcat proporciona un servicio de emulación JNDI para acceder a estos recursos.

Los recursos a los que se accede a través de solicitudes JNDI son dos:

- Un origen de datos JDBC
- Una sesión JavaMail

JDBC es un conocido API estándar que permite a los programadores de aplicaciones acceder a bases de datos relacionales (como MySQL, Oracle y SQL Server) de forma estándar y uniforme. Para acceder a datos de estas bases de datos relacionales, la aplicación debe obtener un objeto DataSource.

JavaMail es otro conocido API que proporciona una interfaz para acceder a funciones de cliente de correo electrónico (es decir, para crear y enviar correo) a través de diferentes métodos de procesar correo electrónico, de forma estándar y

uniforme. Para que una aplicación Web pueda acceder a servidores de correo y enviar correo, primero debe obtener un objeto Session de JavaMail.
Para JNDI, dispones de las siguientes opciones a la hora de configurar el recurso dentro de la jerarquía de los componentes de configuración de Tomcat:

- En el nivel <GlobalNamingResources> global de servidor
- En el nivel <DefaultContext> del host virtual
- En el nivel <Context> asociado a una aplicación Web que habitualmente se encuentra en el archivo XML de descriptor de contexto de la aplicación

Cualquier recurso JNDI configurado en el nivel <DefaultContext> estará disponible para todas las aplicaciones Web que se ejecuten en el mismo host virtual, mientras que cualquier recurso JNDI configurado en el nivel <Context> sólo estará disponible entro de la aplicación Web concreta asociada a dicho contexto. Los recursos configurados en el nivel <GlobalNamingResources> estarán disponibles en todo el servidor (en todos los servicios y motores).
En nuestro caso, se ha configurado un recurso en el nivel <GlobalNamingResources> (usuario y contraseña de aplicaciones manager de tomcat), y dos en el nivel <Context>, correspondientes a la conectividad con la base de datos Oracle y con el servidor de correo.

- Conexión origen de datos JDBC

Project.net procesa datos, que se almacenan en una base de datos Oracle. El sistema de administración de la base de datos Oracle se basa en conceptos relacionales y, por ello, se denomina sistema de administración de base de datos relacional (o RDBMS).
JDBC es una interfaz de programación para acceder a RDBMS. Su funcionamiento se basa en la transmisión y ejecución del Lenguaje estructurado de consultas (SQL) en el servidor de base de datos. SQL es un lenguaje de consultas basado en texto para realizar operaciones con datos almacenados en una base de datos relacional. Las operaciones de JDBC se encargan de lo siguiente:

- Recibir las llamadas del API JDBC y transformarlas en una consulta SQL.
- Enviar la consulta al motor de procesamiento SQL del RDBMS.
- Recuperar el conjunto de resultados devuelto desde la consulta y transformarlo en una estructura de datos accesible para Java.

No todas las instrucciones generan un conjunto de resultados. Si una búsqueda no es satisfactoria, el conjunto de resultados devuelto estará vacío (un conjunto de

resultados NULL). Además, el lenguaje SQL incluye instrucciones que se utilizan para crear tablas, actualizar datos, borrar filas, etc.; estas instrucciones tampoco devuelven conjuntos de resultados.

A continuación se describen los pasos necesarios para configurar recursos JNDI para un origen de datos JDBC:

1. Añada etiquetas <Resource> y <ResourceParams> (ésta si fuese necesario) al elemento <Context> de la aplicación Web o en un subelemento <DefaultContext> del elemento <Host> para configurar el recurso JNDI.

```
<Resource name="jdbc/PnetDB" auth="Container"
type="javax.sql.DataSource" username="pnet_user"
password="pnet_user" driverClassName="oracle.jdbc.OracleDriver"
url="jdbc:oracle:thin:@localhost:1521:XE" maxActive="8" maxIdle="4"
/>
```

Los parámetros de configuración son los siguientes:

- DriverClassName: Nombre de la clase de Java del controladore JDBC.
- maxActive: Número máximo de conexiones de esta agrupación. La agrupación de conexiones se utiliza para reducir el número de conexiones y desconexiones entre consultas. Se crean un grupo de conexiones físicas al inicio del sistema (DBCP) y cuando una aplicación necesita una conexión, se le proporciona una de estas conexiones.
- maxIdle: Número máximo de conexiones latentes en esta agrupación.
- Password: Contraseña utilizada para conectarse a la base de datos.
- Url: El URL compatible con JDBC que especifica la instancia de base de datos que utilizar.
- Username: El ID de usuario utilizado para iniciar sesión en la base de datos.

2. Se define un elemento <resource-ref>, correspondiente al element <Resource> anterior, en el archivo web.xml de Project.net que utilice el recurso JDBC.

```
<resource-ref>
      <description>Default data source</description>
      <res-ref-name>jdbc/PnetDB</res-ref-name>
      <res-type>javax.sql.DataSource</res-type>
      <res-auth>CONTAINER</res-auth>
</resource-ref>
```

3. Se utiliza llamadas JNDI en el código de la aplicación para buscar el origen de datos JDBC.

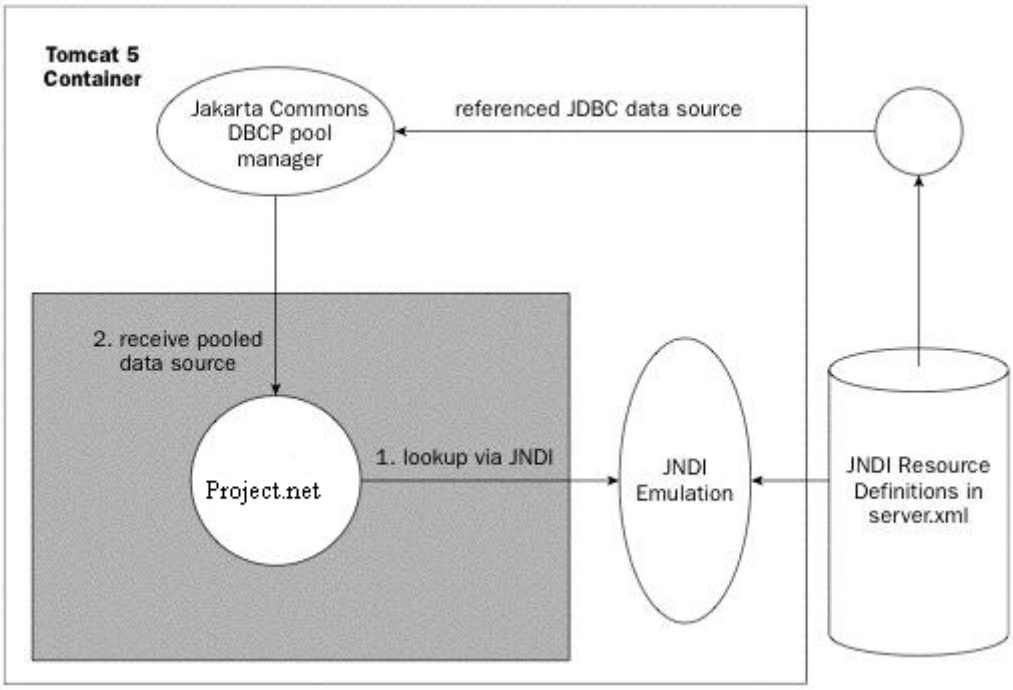


Ilustración 34: Búsqueda del origen de datos por parte de Project.net, extraída de [3]

En el diagrama se describen los siguientes pasos:

- Project.net obtiene un InitialContex JNDI de Tomcat; tras ello, realiza una búsqueda en el recurso (origen de datos JDBC) por nombre.
- Tomcat procesa la búsqueda JNDI consultando los archivos de configuración (server.xml y web.xml) para determinar que controlador JDBC utiliza para obtener un origen de datos. Tomcat también puede utilizar la agrupación de conexiones de bases de datos (DBCP) para agrupar las conexiones realizadas.

- Configurar sesión de correo

Cuando un usuario decide invitar a otro usuario a unirse al proyecto, Project.net envía una invitación al correo del usuario invitado. Para ello, es necesaria la conexión de Project.net a un servidor de correo saliente denominado servidor de correo SMTP. Este servidor de correo puede ser local o remoto.

En nuestro caso, Project.net se conecta al servidor de correo SMTP de GMail. También se instaló uno local, pero el problema que tenía es que, la mayoría de servidores de correo entrante denominado POP3, rechazan correos provenientes de servidores con IP dinámica, y la IP que nos suministra nuestro proveedor de servicios, YA.COM, es dinámica.

JavaMail es un API de programación estándar que los programadores de Java utilizan para crear y enviar correo electrónico. Tomcat admite JavaMail proporcionando la configuración JNDI de una sesión JavaMail como recurso, por medio de su propio código para crear una sesión JavaMail para la aplicación Web. Esto permite a Project.net utilizar JNDI para buscar y utilizar la sesión. El Servlet utiliza la sesión JavaMail configurada para mandar un correo electrónico.
Para configurar la sesión de correo, debemos añadir en el descriptor de contexto de Project.net lo que se muestra a continuación.

```
<Resource name="mail/PnetSession" auth="Container"
type="javax.mail.Session" mail.transport.protocol="smtp"
mail.smtp.auth="true" mail.smtp.host="smtp.gmail.com" mail.smtp.port="465"
mail.smtp.user="2010project.net@gmail.com" password="xxxxxxxx"
mail.smtp.quitwait="false" mail.smtp.starttls.enable="true"
mail.smtp.socketFactory.class="javax.net.ssl.SSLSocketFactory"
mail.debug="true" />
```

De esta forma se configura el contexto JNDI mail/sesión, que hace referencia al servidor SMTP de GMail que utiliza el puerto 465. La cuenta de correo desde la cual se mandan las invitaciones es 2010project.net@gmail.com.
No se nos puede olvidar, declarar una referencia al recurso JNDI en el descriptor de implementación.

```
<resource-ref>
    <description>Default mail session</description>
    <res-ref-name>mail/PnetSession</res-ref-name>
    <res-type>javax.mail.Session</res-type>
    <res-auth>CONTAINER</res-auth>
```

Quedando ya configurada de esta manera la sesión de correo.

4.3.3. Pruebas

4.3.3.1. Pruebas en la máquina principal

En primer lugar se utilizará la versión de evaluación del software JProfiler, que es una utilidad para analizar aplicaciones Java. Analizaremos el uso que hace Tomcat de la CPU y de memoria.
Haremos el siguiente experimento que dura 5 minutos siguiendo estos cuatro pasos:

1. Arrancamos el servidor.
2. En el minuto 2 de nuestro experimento, tecleamos en el navegador http://mariajoproject.dyndns.org:7070.
3. En el minuto 3 iniciamos sesión en Project.net con una cuenta.
4. En los minutos 3.5 y 4 aproximadamente, navegando por Project.net, hacemos click en uno de los botones del panel.

Veamos los resultados.

En la siguiente gráfica se muestra el consumo de la CPU.

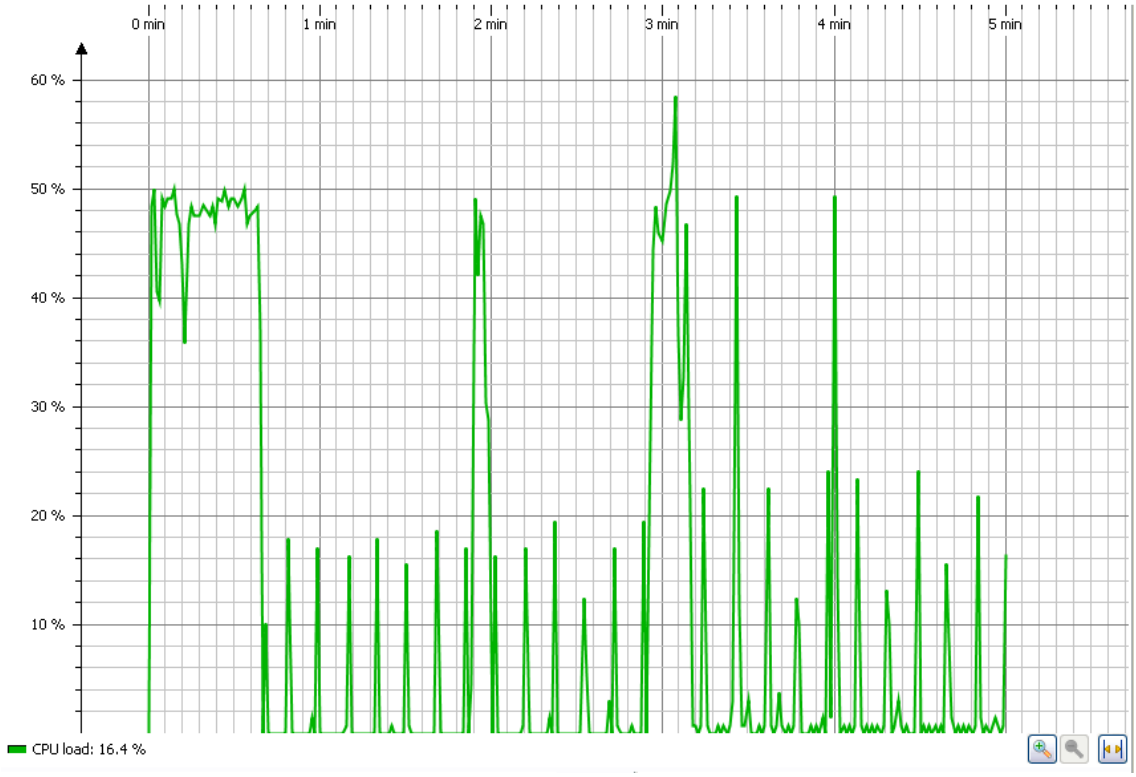


Ilustración 35: Consumo de CPU por parte de Tomcat en la máquina principal

Observamos que pasados 40 segundos aproximadamente se produce una bajada en el consumo de la CPU. Esta bajada se corresponde con el momento en el que el servidor termina de cargarse completamente, como se demuestra en el siguiente dibujo.

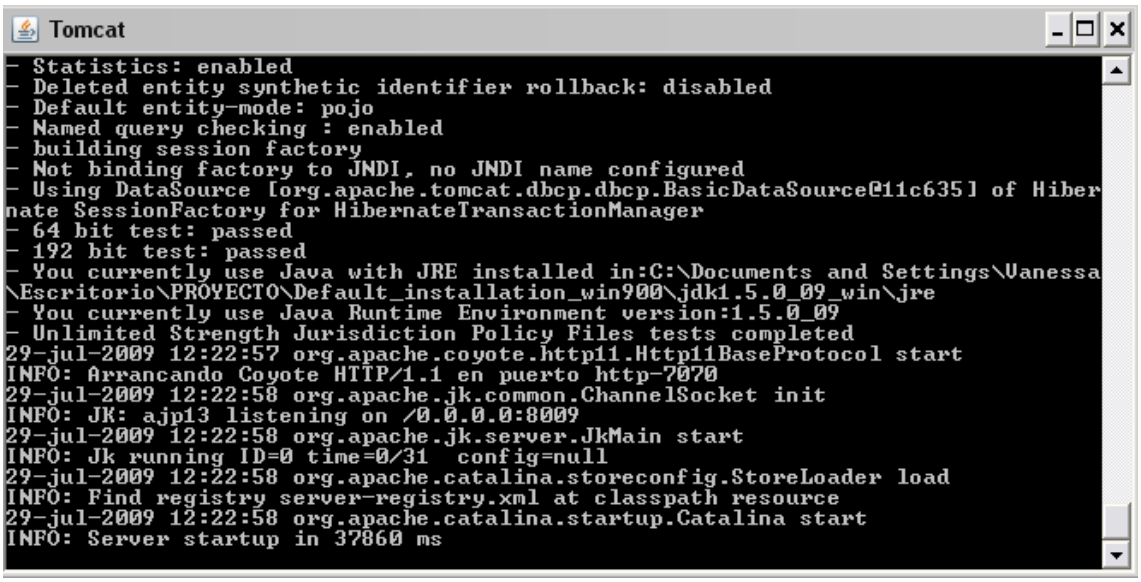


Ilustración 36: Apache Tomcat corriendo en la máquina principal

Observamos también como cuando pasan 2, 3, 3'5 y 4 minutos se produce un pico de consumo correspondiente a los pasos 2, 3, y 4 arriba explicados

En cuanto al uso de memoria, podemos decir que cada vez que se produce un evento se produce un cambio brusco, ya que se liberan objetos, pero al mismo tiempo se utilizan otros. Estos cambios influyen en la actividad del recolector de basura que es el encargado de gestionar el espacio de memoria de forma óptima.

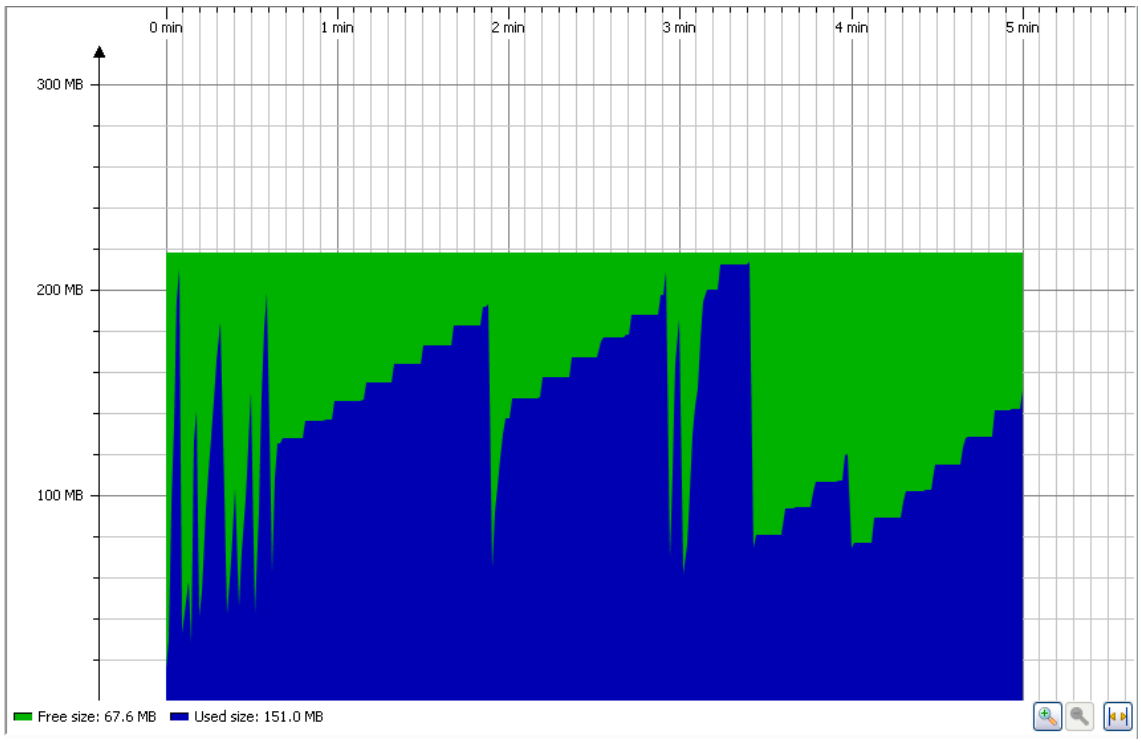


Ilustración 37: Uso de memoria de Tomcat en la máquina principal

Efectivamente, con la siguiente gráfica se corrobora lo dicho unas líneas antes, el recolector de basura tiene sus picos de actividad mientras que se está cargando el servidor, y posteriormente con cada evento.

Ilustración 38: Actividad del recolector de basura en la máquina principal

Una vez analizado el consumo de CPU y de memoria de Project.net, lo someteremos a una carga masiva de usuarios, mediante el software Web Stress Tool.

Suponemos que existen 10 usuarios, y todos ellos quieren conectarse a Project.net.

```
** Test Logfile by Webserver Stress Tool 7.2.2.261 Trial Version **
© 1998-2009 Paessler AG, http://www.paessler.com

Test run on 29/07/2009 13:27:48

** Project and Scenario Comments, Operator **



Results of period #1 (from 0 sec to 10 sec ):
************************************************************
Completed Clicks: 10 with 0 Errors (=0,00%)
Average Click Time for 10 Users: 1.358 ms
```

Successful clicks per Second: 1,05 (equals 3.761,11 Clicks per Hour)

Results of complete test
****

** Results per URL for complete test **

URL#1 (http://mariajoproject.dyndns.org:7070): Average Click Time 1.358 ms, 10 Clicks, 0 Errors

Total Number of Clicks: 10 (0 Errors)
Average Click Time of all URLs: 1.358 ms

Se obtienen los siguientes resultados.

** Test Logfile by Webserver Stress Tool 7.2.2.261 Trial Version **
© 1998-2009 Paessler AG, http://www.paessler.com
Test run on 29/07/2009 13:27:48
** Project and Scenario Comments, Operator **
** Test Setup **
Test Type: CLICKS (run test until 1 clicks per user)
User Simulation: 10 simultaneous users - 5 seconds between clicks
Logging Period: Log every 10 seconds
** URLs **
URL Sequencing: Users always click the same URL (to spreads load evenly on all URLs, set number of users to a multiple of the number of URLs!)
1 URLs
URL#1: GET http://mariajoproject.dyndns.org:7070 POSTDATA= Click Delay=10
** Browser Settings **
Browser Simulation:
User Agent: Mozilla/5.0 (compatible; Webserver Stress Tool 7; Windows)
HTTP Request Timeout: 120 s
Recursive Browsing / HTML Parsing:
** Options **
Advanced Settings:
Logging:
Write detailed log(s)
Timer: not enabled
Local IPs: Automatic
Advanced Data Merging Features:
** Client System **
Windows XP V5.1 (Build 2600) Service Pack 2, CPU Proc. Type 586 (Rev. 27394) at 2.712 MHz, 2006 MB available RAM of 2147 MB total physical RAM, 3117 MB available pagefile, 14297 MB free disk space on C:

```
Test is starting
----------------------------------------------------
Creating Users...
Test preparations done
Master Controller will start now...
Switching to 10 users.
************************************************************
Test Done, now waiting for simulated users to stop surfing
************************************************************
Waiting for 5 seconds for users to cool down and stop surfing...
All surfers passed away...
Master Controller is halted
Results of period #1 (from 0 sec to 10 sec ):
************************************************************
Analyzed Time span of this period: 9.572 msec
Sent/Received Requests: 10 / 11 (=-10,00% not answered in this period)
Completed Hits: 10 (including images, frames etc.)
Completed Clicks: 10 with 0 Errors (=0,00%)
Results for Images for this period:
Image Hits: 0 with 0 Errors
Spectrum of Click Times
0,00% of All Users waited <100ms
0,00% of All Users waited <200ms
10,00% of All Users waited <500ms
70,00% of All Users waited <1s
0,00% of All Users waited <2s
10,00% of All Users waited <5s
10,00% of All Users waited <10s
0,00% of All Users waited <20s
0,00% of All Users waited <50s
0,00% of All Users waited <100s
Measured Times:
Average Time to Connect for 10 Users: 374 ms
Average Time to First Byte for 10 Users: 466 ms
Average Click Time for 10 Users: 1.358 ms
Hits per Second: 1,05 (equals 3.761,11 Hits per Hour)
Successful clicks per Second: 1,05 (equals 3.761,11 Clicks per Hour)
Results per URL for this Period:
URL#1 (http://mariajoproject.dyndns.org:7070): Average Click Time 1.358 ms, 10 Clicks, 0
Errors,
Average Click Time of all URLs: 1.358 ms
Resulting page statuscodes for this period:
10x "200" (=OK)
Datavolume and Bandwidth for this period:
Average User Bandwidth: 150,99 kbit/s
```

```
Total Bytes: 139.854 Bytes (140 kByte) (Throughput ~117 kbit/sec)
Results of complete test
************************************************************
** Results per User for complete test **
User #1: Avg. Click Time: 355,83 ms, 1 Clicks, 1 Hits, 0 Errors, 13.985 Bytes, 314,42 kbit/s
User #2: Avg. Click Time: 536,31 ms, 1 Clicks, 1 Hits, 0 Errors, 13.986 Bytes, 208,63 kbit/s
User #3: Avg. Click Time: 670,85 ms, 1 Clicks, 1 Hits, 0 Errors, 13.985 Bytes, 166,77 kbit/s
User #4: Avg. Click Time: 934,72 ms, 1 Clicks, 1 Hits, 0 Errors, 13.986 Bytes, 119,70 kbit/s
User #5: Avg. Click Time: 812,01 ms, 1 Clicks, 1 Hits, 0 Errors, 13.986 Bytes, 137,79 kbit/s
User #6: Avg. Click Time: 724,78 ms, 1 Clicks, 1 Hits, 0 Errors, 13.985 Bytes, 154,36 kbit/s
User #7: Avg. Click Time: 5.194,39 ms, 1 Clicks, 1 Hits, 0 Errors, 13.985 Bytes, 21,54 kbit/s
User #8: Avg. Click Time: 3.045,24 ms, 1 Clicks, 1 Hits, 0 Errors, 13.986 Bytes, 36,74 kbit/s
User #9: Avg. Click Time: 558,55 ms, 1 Clicks, 1 Hits, 0 Errors, 13.985 Bytes, 200,31 kbit/s
User #10: Avg. Click Time: 747,90 ms, 1 Clicks, 1 Hits, 0 Errors, 13.985 Bytes, 149,59 kbit/s
** Results per URL for complete test **
URL#1 (http://mariajoproject.dyndns.org:7070): Average Click Time 1.358 ms, 10 Clicks, 0
Errors
29/07/2009 13:27:58: Total Number of Clicks: 10 (0 Errors)
29/07/2009 13:27:58: Average Click Time of all URLs: 1.358 ms
```

Se utilizan los siguientes parámetros que se definen a continuación:

- Click: click simulado de ratón de un usuario que envía una petición al servidor.
- Request: Petición HTTP enviada al servidor.
- Hit: Petición HTTP completada (enviada y recibida por parte del servidor=.
- Time to connect: Tiempo que pasa desde que se envía la petición hasta que el cliente se conecta con el servidor.
- Time to first byte: Tiempo que pasa desde que se envía la petición hasta que se recibe por parte del servidor el primer byte.
- Click time: Tiempo que usuario está esperando hasta que su click termina.
- Sent Requests: Número de peticiones enviadas al servidor.
- Received Requests: Número de peticiones recibidas desde el servidor.

Entre todos los datos resultantes de la simulación destacamos que el tiempo medio de conexión al servidor es de 374 milisegundos. Podemos decir, que ante esta carga el servidor responde correctamente.

Vemos lo que ocurre si repetimos el mismo experimento, es decir, 10 usuarios intentan conectarse a http://mariajoproject.dyndns.org:7070 pero en este caso no tenemos el servidor en marcha.

```
** Test Logfile by Webserver Stress Tool 7.2.2.261 Trial Version **
© 1998-2009 Paessler AG, http://www.paessler.com
```

```
Test run on 29/07/2009 14:54:22
** Project and Scenario Comments, Operator **
** Test Setup **
Test Type: CLICKS (run test until 1 clicks per user)
User Simulation: 10 simultaneous users - 5 seconds between clicks
Logging Period: Log every 10 seconds
** URLs **
URL Sequencing: Users always click the same URL (to spreads load
evenly on all URLs, set number of users to a multiple of the number of
URLs!)
URLs
URL#1: GET http://mariajoproject.dyndns.org:7070 POSTDATA= Click
Delay=10
** Browser Settings **
Browser Simulation:
User Agent: Mozilla/5.0 (compatible; Webserver Stress Tool 7; Windows)
HTTP Request Timeout: 120 s
Recursive Browsing / HTML Parsing:
** Options **
Advanced Settings:
Logging:
Write detailed log(s)
Timer: not enabled
Local IPs: Automatic
Advanced Data Merging Features:
** Client System **
Windows XP V5.1 (Build 2600) Service Pack 2, CPU Proc. Type 586 (Rev.
27394) at 2.712 MHz,
2147 MB available RAM of 2147 MB total physical RAM, 3550 MB available
pagefile, 14275 MB free disk space on C:
Test is starting
-----------------------------------------------------
Creating Users...
Test preparations done
Master Controller will start now...
Switching to 10 users.
User #1 : !FAILED! CLICK-Request 1: Time=1.211 ms, TFB=0 ms, Bytes=0,
HTTP-StatusCode=404 (Not Found) Socketcode = 10061 (Connection
refused.) URL #1=http://mariajoproject.dyndns.org:7070 Connection
refused (Error #10061)
User #2 : !FAILED! CLICK-Request 1: Time=1.080 ms, TFB=0 ms, Bytes=0,
HTTP-StatusCode=404 (Not Found) Socketcode = 10061 (Connection
refused.) URL #1=http://mariajoproject.dyndns.org:7070 Connection
refused (Error #10061)
User #3 : !FAILED! CLICK-Request 1: Time=1.087 ms, TFB=0 ms, Bytes=0,
HTTP-StatusCode=404 (Not Found) Socketcode = 10061 (Connection
refused.) URL #1=http://mariajoproject.dyndns.org:7070 Connection
refused (Error #10061)
User #4 : !FAILED! CLICK-Request 1: Time=949 ms, TFB=0 ms, Bytes=0,
HTTP-StatusCode=404 (Not Found) Socketcode = 10061 (Connection
refused.) URL #1=http://mariajoproject.dyndns.org:7070 Connection
refused (Error #10061)
User #5 : !FAILED! CLICK-Request 1: Time=990 ms, TFB=0 ms, Bytes=0,
HTTP-StatusCode=404 (Not Found) Socketcode = 10061 (Connection
refused.) URL #1=http://mariajoproject.dyndns.org:7070 Connection
refused (Error #10061)
User #6 : !FAILED! CLICK-Request 1: Time=977 ms, TFB=0 ms, Bytes=0,
HTTP-StatusCode=404 (Not Found) Socketcode = 10061 (Connection
refused.) URL #1=http://mariajoproject.dyndns.org:7070 Connection
refused (Error #10061)
User #7 : !FAILED! CLICK-Request 1: Time=993 ms, TFB=0 ms, Bytes=0,
```

```
HTTP-StatusCode=404 (Not Found) Socketcode = 10061 (Connection
refused.) URL #1=http://mariajoproject.dyndns.org:7070 Connection
refused (Error #10061)
User #8 : !FAILED! CLICK-Request 1: Time=973 ms, TFB=0 ms, Bytes=0,
HTTP-StatusCode=404 (Not Found) Socketcode = 10061 (Connection
refused.) URL #1=http://mariajoproject.dyndns.org:7070 Connection
refused (Error #10061)
User #9 : !FAILED! CLICK-Request 1: Time=987 ms, TFB=0 ms, Bytes=0,
HTTP-StatusCode=404 (Not Found) Socketcode = 10061 (Connection
refused.) URL #1=http://mariajoproject.dyndns.org:7070 Connection
refused (Error #10061)
User #10 : !FAILED! CLICK-Request 1: Time=964 ms, TFB=0 ms, Bytes=0,
HTTP-StatusCode=404 (Not Found) Socketcode = 10061 (Connection
refused.) URL #1=http://mariajoproject.dyndns.org:7070 Connection
refused (Error #10061)
**********************************************************************
Test Done, now waiting for simulated users to stop surfing
**********************************************************************
Waiting for 5 seconds for users to cool down and stop surfing...
All surfers passed away...
Master Controller is halted
Results of period #1 (from 1 sec to 8 sec ):
**********************************************************************
Analyzed Time span of this period: 7.257 msec
Sent/Received Requests: 10 / 10 (=0,00% not answered in this period)
Completed Hits: 10 (including images, frames etc.)
Completed Clicks: 10 with 10 Errors (=100,00%)
Results for Images for this period:
Image Hits: 0 with 0 Errors
Measured Times:
Hits per Second: 1,38 (equals 4.960,75 Hits per Hour)
Successful clicks per Second: 0,00 (equals 0,00 Clicks per Hour)
Results per URL for this Period:
URL#1 (http://mariajoproject.dyndns.org:7070): 10 Clicks, 10 Errors,
Average Click Time of all URLs: 0 ms
Resulting page statuscodes for this period:
0x "200" (=OK)
10x "404" (=Not Found)
Datavolume and Bandwidth for this period:
Average User Bandwidth: 0,00 kbit/s
29/07/2009 14:54:31: Total Bytes: 0 Bytes (0 kByte) (Throughput ~0
kbit/sec)
Results of complete test
**********************************************************************
** Results per User for complete test **
User #1: Avg. Click Time: 1.235,62 ms, 1 Clicks, 1 Hits, 1 Errors, 0
Bytes, 0,00 kbit/s
User #2: Avg. Click Time: 1.087,85 ms, 1 Clicks, 1 Hits, 1 Errors, 0
Bytes, 0,00 kbit/s
29/07/2009 14:54:31: User #3: Avg. Click Time: 1.087,07 ms, 1 Clicks,
1 Hits, 1 Errors, 0 Bytes, 0,00 kbit/s
29/07/2009 14:54:31: User #4: Avg. Click Time: 949,36 ms, 1 Clicks, 1
Hits, 1 Errors, 0 Bytes, 0,00 kbit/s
29/07/2009 14:54:31: User #5: Avg. Click Time: 990,21 ms, 1 Clicks, 1
Hits, 1 Errors, 0 Bytes, 0,00 kbit/s
29/07/2009 14:54:31: User #6: Avg. Click Time: 977,42 ms, 1 Clicks, 1
Hits, 1 Errors, 0 Bytes, 0,00 kbit/s
29/07/2009 14:54:31: User #7: Avg. Click Time: 993,47 ms, 1 Clicks, 1
Hits, 1 Errors, 0 Bytes, 0,00 kbit/s
29/07/2009 14:54:31: User #8: Avg. Click Time: 973,57 ms, 1 Clicks, 1
Hits, 1 Errors, 0 Bytes, 0,00 kbit/s
```

```
User #9: Avg. Click Time: 987,45 ms, 1 Clicks, 1 Hits, 1 Errors, 0
Bytes, 0,00 kbit/s
User #10: Avg. Click Time: 963,86 ms, 1 Clicks, 1 Hits, 1 Errors, 0
Bytes, 0,00 kbit/s
Results per URL for complete test **
URL#1 (http://mariajoproject.dyndns.org:7070): 10 Clicks, 10 Errors
Total Number of Clicks: 10 (10 Errors)
Average Click Time of all URLs: 0 ms
```

El resultado era el esperado. El navegador de estos usuarios ficticios les devolvería el código de respuesta 404, que es el mensaje que devuelve el protocolo HTTP cuando el servidor no se encuentra.

4.3.3.2. Pruebas en la máquina virtual

Volvemos a repetir los experimentos, pero esta vez con Project.net instalada dentro de la máquina virtual.

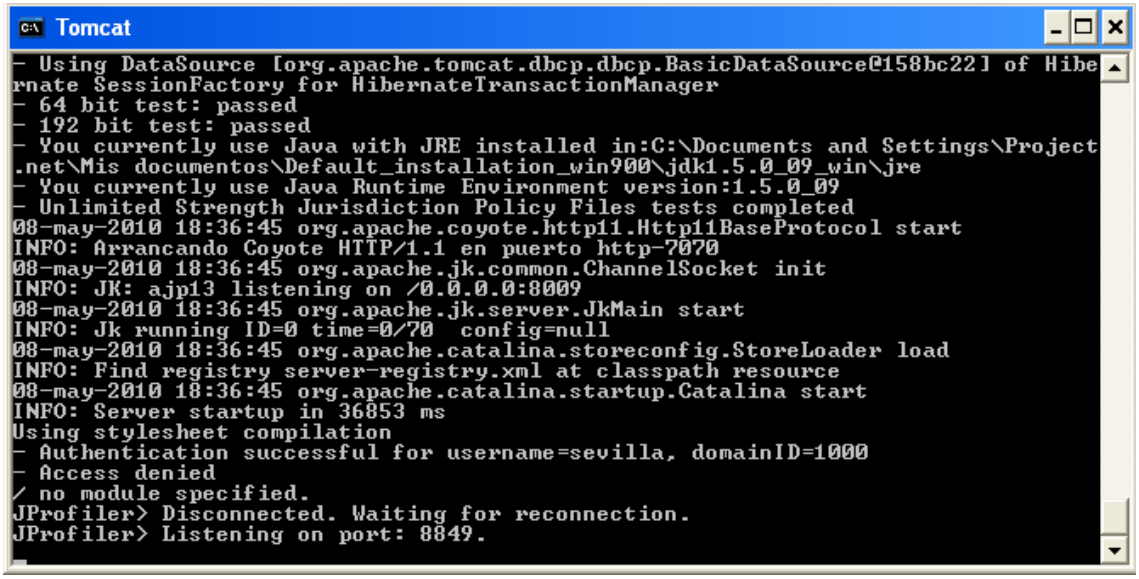


Ilustración 39: Apache Tomcat corriendo en la máquina virtual

Tomcat tarda en arrancarse 36,853 segundos. Siendo este periodo, el de mayor consumo de CPU.

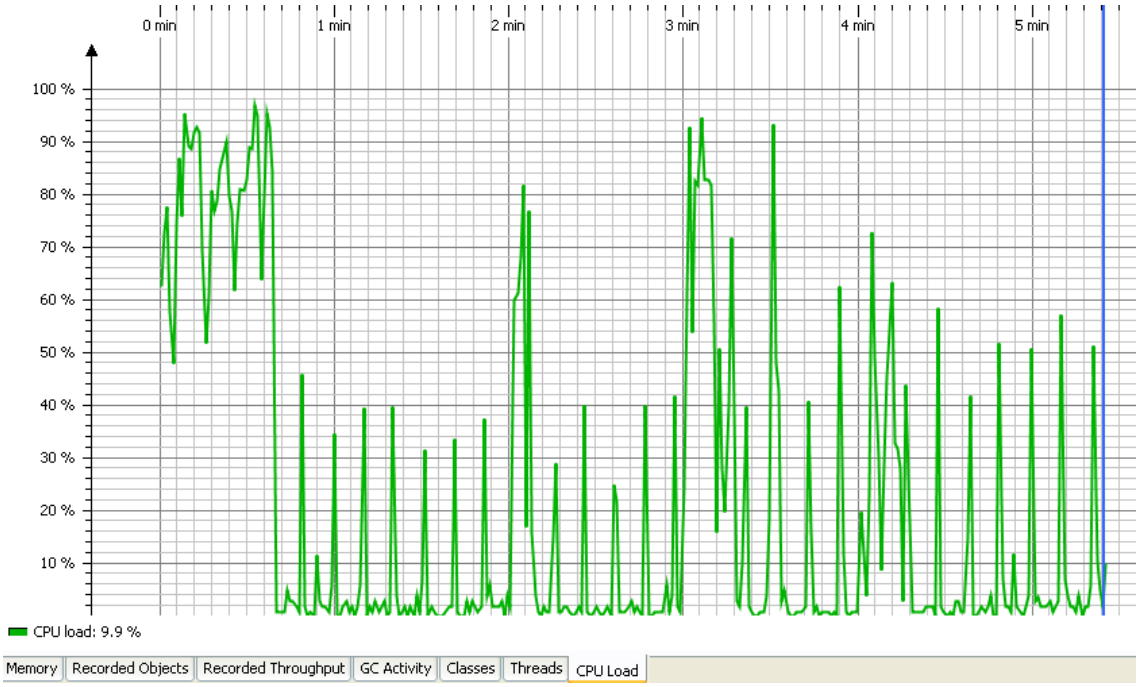


Ilustración 40: Consumo de CPU de Apache Tomcat en la máquina virtual

El tiempo que tarda en cargarse es similar en ambas máquinas, pero la diferencia significativa entre los dos experimentos está en que, mientras en la primera máquina, no se llega a consumir el 50% de los recursos de la CPU, en la máquina virtual, el consumo medio está en torno al 80% durante esos 36 segundos. Posteriormente, al loguearnos y al navegar (hacer click) por Project.net, el consumo de CPU sería similar, en torno al 80% de nuevo.

La causa de estas diferencias, entre la máquina 1 y la virtual, está en la diferencia de memoria RAM, pasamos de 2,87 Gigas de RAM a 512 Megas.

En cuanto al uso de memoria, lógicamente los resultados del experimento son similares a los resultados que se obtuvieron con la máquina 1, ya que el número de clases y objetos que deben cargarse es exactamente el mismo, en un caso que en otro.

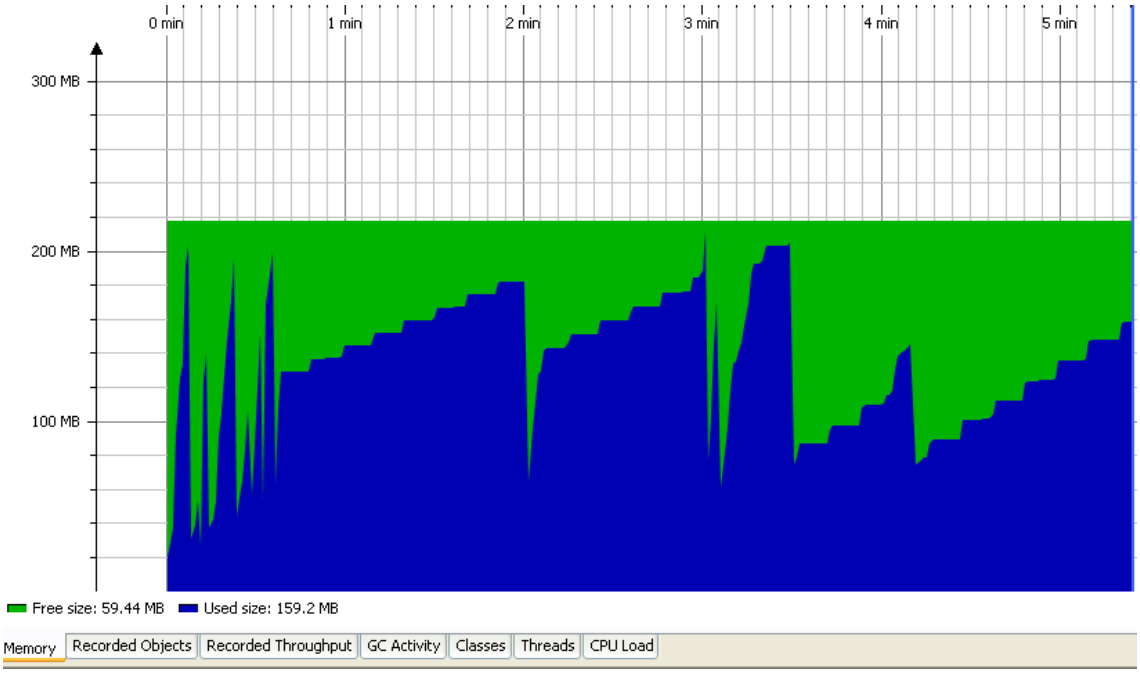


Ilustración 41: Uso de memoria de Apache Tomcat en la máquina virtual

Por último, vemos la actividad del recolector, similar también al experimento anterior, puesto que va en función del uso de memoria que se haga, produciéndose sus picos de actividad cada vez que se produce un evento.

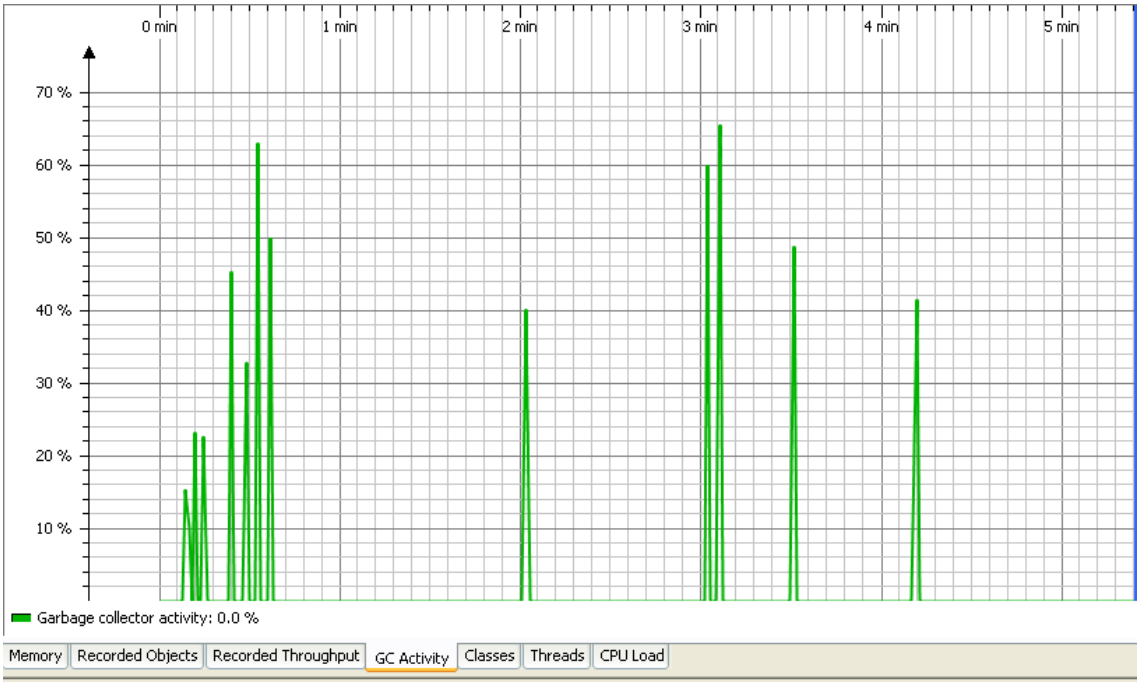


Ilustración 42: Actividad del recolector de basura en la máquina virtual

Realizamos ahora el segundo experimento.

Configuramos Webserver Stress Tool con los mismos parámetros que cuando lo realizamos en la primera máquina. Volvemos a suponer que existen 10 usuarios, y todos ellos quieren acceder a Project.net.

```
** Test Logfile by Webserver Stress Tool 7.2.2.261 Trial Version **
© 1998-2009 Paessler AG, http://www.paessler.com
Test run on 09/05/2010 12:54:51
** Project and Scenario Comments, Operator **
** Test Setup **
Test Type: CLICKS (run test until 1 clicks per user)
User Simulation: 10 simultaneous users - 5 seconds between clicks
Logging Period: Log every 10 seconds
** URLs **
URL Sequencing: Users always click the same URL (to spreads load evenly on all URLs, set
number of users to a multiple of the number of URLs!)
URLs
URL#1: GET project-net:7070 POSTDATA= Click Delay=
Browser Settings **
Browser Simulation:
User Agent: Mozilla/5.0 (compatible; Webserver Stress Tool 7; Windows)
HTTP Request Timeout: 120 s
Recursive Browsing / HTML Parsing:
** Options **
Advanced Settings:
Logging:
Write detailed log(s)
Timer: not enabled
Local IPs: Automatic
Advanced Data Merging Features:
** Client System **
Windows XP V5.1 (Build 2600) Service Pack 2, CPU Proc. Lev. 686 (Rev. 9474) at 2.191 MHz,
09/05/2010 12:54:51:   48 MB available RAM of 536 MB total physical RAM, 655 MB available
pagefile, 3365 MB free disk space on C:
Test is starting
---------------------------------------------------
Creating Users...
Test preparations done
Master Controller will start now...
Switching to 10 users.
************************************************************************
Test Done, now waiting for simulated users to stop surfing
************************************************************************
Waiting for 5 seconds for users to cool down and stop surfing...
All surfers passed away...
Master Controller is halted
Results of period #1 (from 1 sec to 7 sec ):
************************************************************
Analyzed Time span of this period: 6.095 msec
Sent/Received Requests: 10 / 10 (=0,00% not answered in this period)
Completed Hits: 10 (including images, frames etc.)
```

```
Completed Clicks: 10 with 0 Errors (=0,00%)
Results for Images for this period:
Image Hits: 0 with 0 Errors
Spectrum of Click Times
80,00% of All Users waited <100ms
20,00% of All Users waited <200ms
0,00% of All Users waited <500ms
0,00% of All Users waited <1s
0,00% of All Users waited <2s
0,00% of All Users waited <5s
0,00% of All Users waited <10s
0,00% of All Users waited <20s
0,00% of All Users waited <50s
0,00% of All Users waited <100s
Measured Times:
Average Time to Connect for 10 Users: 12 ms
Average Time to First Byte for 10 Users: 74 ms
Average Click Time for 10 Users: 83 ms
Hits per Second: 1,64 (equals 5.906,07 Hits per Hour)
Successful clicks per Second: 1,64 (equals 5.906,07 Clicks per Hour)
Results per URL for this Period:
URL#1 (): Average Click Time 83 ms, 10 Clicks, 0 Errors,
Average Click Time of all URLs: 83 ms
Resulting page statuscodes for this period:
10x "200" (=OK)
Datavolume and Bandwidth for this period:
Average User Bandwidth: 1.378,98 kbit/s
Total Bytes: 139.709 Bytes (140 kByte) (Throughput ~183 kbit/sec)
Results of complete test
************************************************************
```

Se observa en este caso que el tiempo medio de conexión es de 12 ms, muy bajo, dato justificado ya que en este experimento llamamos a Project-net:7070, siendo Project-net, el nombre del host local, es decir, el tiempo que anteriormente se iba en traducir el dominio dyndns a dirección ip, ahora no existe, y la conexión sólo es local.

Repetimos de nuevo el experimento, pero en este caso, suponiendo que existen 5 usuarios.

```
Results of period #1 (from 1 sec to 7 sec ):
************************************************************
Analyzed Time span of this period: 5.831 msec
Sent/Received Requests: 5 / 5 (=0,00% not answered in this period)
Completed Hits: 5 (including images, frames etc.)
Completed Clicks: 5 with 0 Errors (=0,00%)

Measured Times:
Average Time to Connect for 5 Users: 11 ms
Average Time to First Byte for 5 Users: 78 ms
```

Average Click Time for 5 Users: 88 ms

El tiempo medio de conexión pasa de 12 ms a 11 ms, un cambio insignificante. Hubiera sido más interesante, haber hecho el experimento con un número de usuarios mucho mayor, y haber comprobado si se produce una diferencia importante, pero ya versión trial de este software nos limita a un máximo de 10 usuarios.

4.4. Aplicaciones Afines

4.4.1. Redmine

Redmine es una herramienta de gestión de proyectos software con interfaz web. Pertenece, al igual que Project.net, a ese conjunto de herramientas denominadas PPM. Una vez instalada, el administrador puede dar de alta a los desarrolladores y jefes de proyecto (o pueden darse de alta ellos mismos a través de la interface web) [7].

Una vez dados de alta los proyectos y sus jefes, esto pueden definir los hitos del proyecto y las tareas a realizar para cada uno de estos hitos [9]. Si se molesta en meter fechas previstas de inicio y fin de cada tarea, puede obtener el gráfico de Gannt para dicho hito. Si no se desea hacer eso, no es obligatorio, simplemente tendrá una lista de tareas a realizar. Cada tarea se puede asignar a uno de los desarrolladores.

Los desarrolladores tienen en su página de entrada una lista de las tareas que tienen asignadas. Es una única lista conjunta de las tareas de todos los proyectos. Según van trabajando en las tareas, pueden ir marcando el tiempo que estiman que les llevará la tarea, el tiempo que han trabajado en ella y/o el porcentaje que creen que tienen realizado.

Con esta información, en el hito correspondiente del proyecto se muestra una "barra de progreso" horizontal, en la que una parte aparece en color verde, indicando el número de tareas terminadas, mientras que el resto aparece sin color, indicando lo que queda pendiente. Esta barra de progreso da una idea bastante aproximada de cuánto llevamos hecho y cuánto queda por hacer. Por supuesto, será más aproximada si nos molestamos en meter los tiempos estimados en las tareas y estimamos bien.

Una vez que comienzan las pruebas del software, en Redmine también se pueden dar de alta los "bugs" o errores que se encuentren, asignándoselos al desarrollador

correspondiente y al hito para el que consideremos que debe estar corregido dicho error.

Por supuesto, además de todo esto, hay muchas más posibilidades, como:

- Wiki por proyecto.
- Foro por proyecto.
- Envío automático de e-mail a los desarrolladores cada vez que se les asigna una tarea o ante cualquier evento relacionado con el proyecto.
- Posibilidad de subir ficheros y documentos, bien al proyecto, bien como adjuntos a las tareas y errores.
- Posibilidad de definir nuevos tipos de tareas y errores, con campos personalizados, todo ello fácilmente a través de la interface web. Estas tareas personalizadas y campos personalizados se asignan por proyecto, por lo que unos proyectos pueden tener algunas de esas tareas y campos y otros no.
- Se puede ver a través de Redmine los cambios en el repositorio, éste es el lugar donde se almacenan los datos actualizados e históricos de un servidor.
- Entiende CVS, Subversion y algunos de los sistemas de control de versiones más conocidos.
- Gráficos de Gantt, consultas por filtro con posibilidad de guardar esas consultas y proyectos con subproyectos.

Tras ver las funcionalidades más relevantes de Redmine, se va a describir su arquitectura y los modelos de programación que utiliza.

Redmine está hecho sobre Ruby on Rails. Ruby on Rails es un framework, es decir, un conjunto de programas y librerías, aplicaciones web de código abierto escrito en el lenguaje de programación Ruby, siguiendo el paradigma de la arquitectura Modelo Vista Controlador (MVC) [15]. Trata de combinar la simplicidad con la posibilidad de desarrollar aplicaciones del mundo real escribiendo menos código que con otros frameworks y con un mínimo de configuración. El lenguaje de programación Ruby permite la metaprogramación de la cual Rails hace uso, lo que resulta en una sintaxis que muchos de sus usuarios encuentran muy legible. Rails se distribuye a través de RubyGems, que es el formato oficial de paquete y canal de distribución de bibliotecas y aplicaciones Ruby.

Por tanto, para instalar Redmine, es necesario que antes se instalen también Ruby on Rails, Ruby Gems y Rake, éste último consiste en un simple programa con capacidad "make".

Al igual que Project.net, Redmine también necesitará una base de datos previamente instalada. Las bases de datos compatibles con Redmine son:

- MySQL 4.1 o superior.

- PostgreSQL 8.
- SQLITE 3.

Redmine trabaja con el servidor WEBrick de Rails, incluido en Ruby, que consiste en un simple servidor web haciendo uso del protocolo HTTP.

Se muestra a continuación algunas de las captura de pantalla recogidas en la web oficial de Redmine.

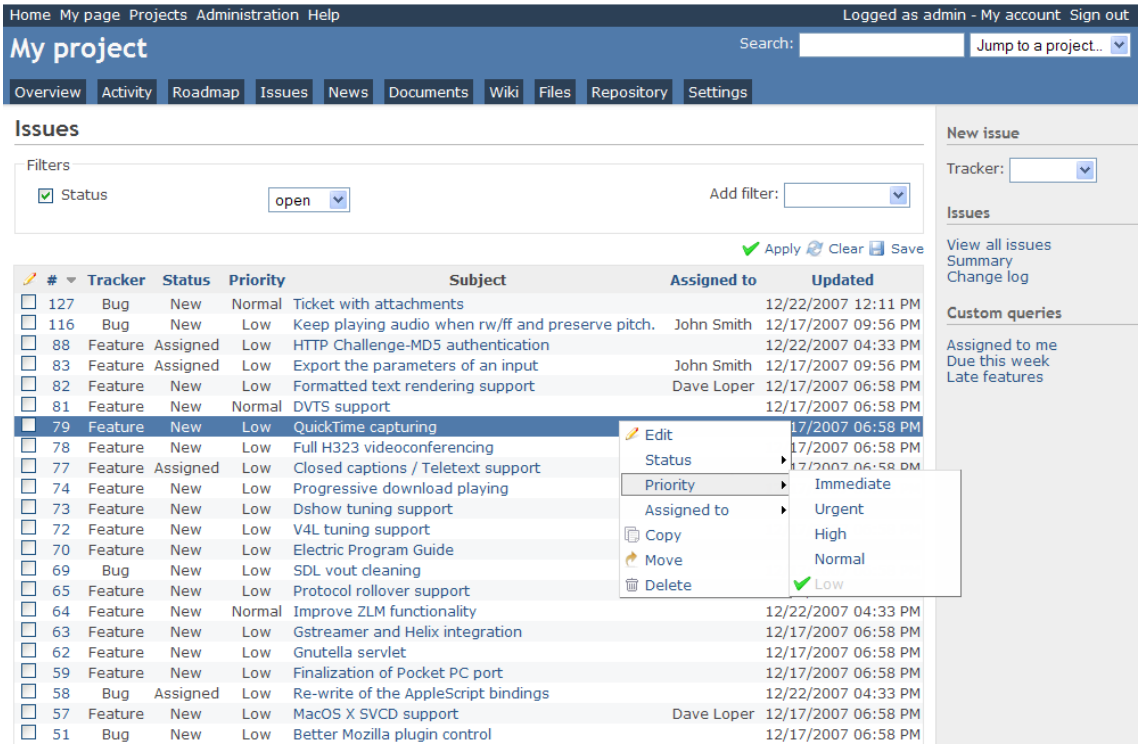


Ilustración 43: Readmine (I), extraída de [7]

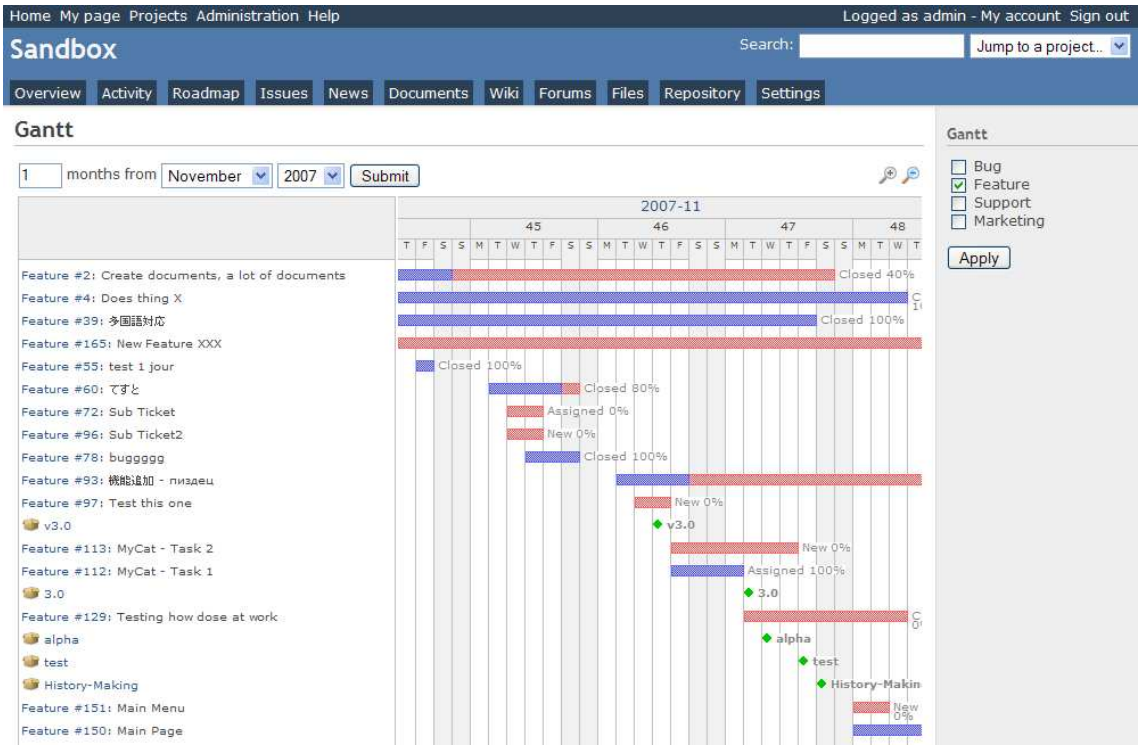


Ilustración 44: Readmine (II), extraída de [7]

Hay que añadir que Ruby on Rails es compatible con el servidor Apache, por lo que, si en algún momento WEBrick se quedara corto dependiendo del tráfico que soporte, podríamos hacer trabajar Redmine con Apache.

4.4.2. EPM Microsoft

Microsoft Project (o MSP) es un software de administración de proyectos diseñado, desarrollado y comercializado por Microsoft para asistir a administradores de proyectos en el desarrollo de planes, asignación de recursos a tareas, dar seguimiento al progreso, administrar presupuesto y analizar cargas de trabajo [5].
La aplicación crea calendarización de rutas críticas, además de cadenas críticas y metodología de eventos en cadena disponibles como add-ons de terceros. Los calendarios pueden ser resource leveled, y las gráficas visualizadas en una Gráfica de Gantt. Adicionalmente, Project puede reconocer diferentes clases de usuarios, los cuales pueden contar con distintos niveles de acceso a proyectos, vistas y otros datos. Los objetos personalizables como calendarios, vistas, tablas, filtros y campos, son almacenados en un servidor que comparte la información a todos los usuarios.

Microsoft Project es una aplicación que por sí sola no permite trabajar de forma colaborativa. Necesitará trabajar con otras herramientas, para alcanzar las funcionalidades de Project.net o Redmine [6].

Microsoft Project y Project Server son piezas angulares de Microsoft Office Enterprise Project Management (EPM).

Pasamos ahora a definir EPM y sus componentes.

La solución EPM (Enterprise Project Management) de Microsoft Office es un entorno de proyecto completo de colaboración y carteras. Esta solución ayuda a la organización a obtener más visibilidad, información detallada y control en todos los trabajos, a mejorar la toma de decisiones, la sintonía con la estrategia empresarial, a maximizar el uso de los recursos, así como a medir y ayudar a incrementar la eficacia operacional.

La solución EPM de Office incluye los siguientes productos de la familia Microsoft Office Project 2007 para proporcionar a las organizaciones una solución completa de administración de las carteras de los proyectos (PPM, Project Portfolio Management):

- Microsoft Office Project Professional: Incluye todas las funcionalidades de Microsoft Office Project Standard, así como unas sólidas herramientas de administración de proyectos con la dosis adecuada de funcionalidad, potencial y flexibilidad, con el fin de administrar los proyectos con mayor eficacia. Puede mantenerse informado, controlar el trabajo del proyecto, las programaciones y las finanzas, así como mantener la sintonía entre los equipos de proyecto, al tiempo que aumenta la productividad gracias a las eficaces opciones de creación de informes, a la integración con los conocidos programas de Microsoft Office system, al planeamiento asistido, a los asistentes y las plantillas. Además, Office Project Professional incluye funcionalidades EPM cuando se está conectado a Microsoft Office Project Server.
- Microsoft Office Project Server: Es la plataforma flexible compatible con las funcionalidades de administración de recursos, programación, elaboración de informes y de colaboración de la solución EPM de Office. Office Project Server permite que las organizaciones almacenen la información de proyectos y recursos central y coherentemente. Asimismo se integra con las funcionalidades de administración de archivos y de colaboración de Microsoft Windows SharePoint Services, ayudando a los integrantes del equipo a trabajar juntos más eficazmente. Además, basándose en estas funciones, los usuarios pueden tener acceso a los datos y a la funcionalidad a través de Internet mediante Microsoft Office Project Web Access.

- Microsoft Office Project Portfolio Server: Es una solución de administración vertical descendente de carteras que ayuda a las organizaciones a tomar conciencia de su potencial identificando, seleccionando, administrando y proporcionando carteras que estén en consonancia con su estrategia empresarial. Office Project Portfolio Server se integra con Office Project Server para proporcionar a las organizaciones una solución completa de administración de las carteras de los proyectos, a las que se tiene acceso a través de Microsoft Office Project Portfolio Web Access.
- Microsoft Office Project Web Access: Es el cliente Web para Office Project Server ofrece a los usuarios una interfaz Web simple para obtener acceso a una serie de funciones de Office Project. Con Office Project Web Access, los usuarios pueden ver, analizar e informar sobre la información del proyecto, incluidos partes de horas, así como crear propuestas de proyectos y planes de actividades. Las funciones de Office Project Web Access se pueden exponer o quitar según la función y el nivel de autorización de un usuario. Office Project Web Access también incluye la interfaz de usuario administrativo. El uso de Office Project Web Access requiere una licencia de acceso de cliente (CAL) de Office Project Server.

Los objetivos de la solución EPM de Microsoft son:

- Normalizar y comunicar un marco de administración en toda la organización.
- Consolidar los datos fundamentales para todas las inversiones empresariales y de TI en un repositorio (servidor) centralizado.
- Proporcionar eficazmente proyectos y carteras de programas que ayuden a maximizar el rendimiento de la inversión.
- Obtener transparencia y control en todas las carteras de aplicaciones existentes.
- Administrar el trabajo desde propuestas sencillas hasta programas de proyectos complejos.
- Dar prioridades, optimizar y seleccionar objetivamente la cartera de proyectos que se adecua a su estrategia empresarial y maximizar el rendimiento de la inversión.
- Predecir proactivamente los costos, los recursos y los retrasos de la programación mediante indicadores clave de rendimiento (KPI, Key Performance Indicators). Se tratan de las métricas financieras o no financieras, utilizadas para cuantificar objetivos que reflejan el rendimiento de una organización, y que generalmente se recogen en su plan estratégico.

- Crear vistas personalizadas (como paneles o tarjetas de puntuación) e informes para obtener transparencia en todos los proyectos, programas y carteras de aplicaciones.
- Analizar la información y, a continuación, usar informes predefinidos o crear un informe personalizado para exponer la información relacionada con el proyecto.
- Colaborar y compartir la información fundamental sin esfuerzo a través de las áreas de trabajo del proyecto del equipo con Windows SharePoint Services.
- Mantener la sintonía de los equipos con las asignaciones de tareas y la información de los partes de horas.
- Iniciar, planear y realizar el seguimiento de los proyectos confidencialmente tanto dentro como fuera de la oficina.
- Usar la integración integrada para comunicar la información relacionada con el proyecto mediante las aplicaciones de Microsoft Office.
- Usar todas las funcionalidades de Project Web Access a través de su exposición como servicios Web.
- Desarrollar soluciones personalizadas con una interfaz de programación de aplicaciones (API) basada en Microsoft .NET Framework y el motor de programación de servidor.

Se va a explicar a continuación, con más detalle, dos de las funcionalidades que diferencia a la solución Microsoft de muchas otras, y que pueden ser beneficiosas para una empresa.

La primera de ellas es la forma que utiliza Microsoft para estimar la duración de un proyecto. Normalmente se trabaja con una estimación única, pero Microsoft hace uso del método PERT. Con este método, en lugar de proporcionar una estimación única para cada tarea, se proporcionarán tres, dando lugar a tres escenarios:

1) Escenario optimista. Ejecución sin materializarse riesgos.
2) Escenario más probable. Ejecución donde se produce una materialización de los riesgos más probables.
3) Escenario pesimista. Se presentan los riesgos que pueden provocar más atraso en el proyecto.

Se deberán añadir a cada tarea, tres estimaciones y el peso de cada estimación.

	Nombre de tarea	Duración	Dur. optimista	Dur. esperada	Dur. pesimista
1	**PROYECTO TEST**	**9 días**	**0 días**	**0 días**	**0 días**
2	Tarea 1	2 días	2 días	3 días	5 días
3	Tarea 2	1 día	1 día	2 días	3 días
4	Tarea 3	1 día	1 día	2 días	3 días
5	Tarea 4	1 día	1 día	2 días	4 días
6	Tarea 5	3 días	3 días	4 días	7 días
7	Tarea 6	2 días	2 días	3 días	5 días

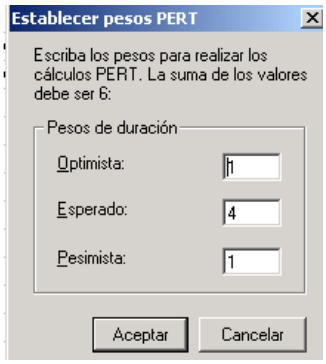


Ilustración 45: Método Pert en EPM Microsoft, extraída de [11]

Al evaluar estos riesgos, la duración del proyecto aumentará automáticamente.

En las dos próximas figuras se ve el tiempo estimado inicialmente, y el tiempo estimado tras la utilización del método PERT.

	Nombre de tarea	Duración	Comienzo	Fin
1	**PROYECTO TEST**	**9 días**	**lun 02/02/04**	**jue 12/02/04**
2	Tarea 1	2 días	lun 02/02/04	mar 03/02/04
3	Tarea 2	1 día	mié 04/02/04	mié 04/02/04
4	Tarea 3	1 día	mié 04/02/04	mié 04/02/04
5	Tarea 4	1 día	jue 05/02/04	jue 05/02/04
6	Tarea 5	3 días	vie 06/02/04	mar 10/02/04
7	Tarea 6	2 días	mié 11/02/04	jue 12/02/04

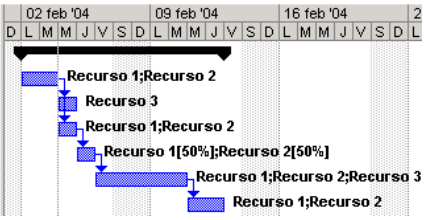


Ilustración 46: Diagrama de Gantt sin riesgos, extraída de [11]

	Nombre de tarea	Duración	Comienzo	Fin
1	**PROYECTO TEST**	**14,83 días**	**lun 02/02/04**	**vie 20/02/04**
2	Tarea 1	3,17 días	lun 02/02/04	jue 05/02/04
3	Tarea 2	2 días	jue 05/02/04	lun 09/02/04
4	Tarea 3	2 días	jue 05/02/04	lun 09/02/04
5	Tarea 4	2,17 días	lun 09/02/04	mié 11/02/04
6	Tarea 5	4,33 días	mié 11/02/04	mar 17/02/04
7	Tarea 6	3,17 días	mar 17/02/04	vie 20/02/04

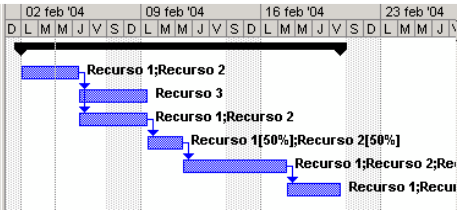


Ilustración 47: Diagrama de Gantt con riesgos, extraída de [11]

La segunda funcionalidad que se quiere destacar de la solución EPM de Microsoft, es el tratamiento que hace de los costes, con el objetivo de optimizar el presupuesto final del proyecto. Para llegar hasta ese objetivo, se puede:

1) Revisar los datos del historial.

Examinar el historial de costes de proyectos antiguos, permite estimar mejor los costes del proyecto actual.

2) Insertar información de costes.

Se pueden hacer estimaciones de costes, insertando los costes de recursos, de tareas o de ambos.

3) Revisar los costes programados.

Una vez insertados los costes, se pueden examinar para ver si necesita ajustarlos para cumplir sus objetivos de costes. Como ya se vio anteriormente, se podía crear

tres escenarios con el método PERT evaluando los riesgos, es posible estimar los costes para cada escenario.

4) Optimización de costes.

Antes de establecer la estimación del coste final, pueden hacerse ajustes finales, para que esa estimación sea lo más realista posible. Se podría reemplazar una asignación de recursos, o quitarla, o incluso, se podrían reducir los costes de una tarea, por ejemplo, reduciendo los costes de viaje.

Por tanto, con estos cuatro pasos, habríamos planificado el proyecto de una u otra forma, con el objetivo final de conseguir el menor coste.

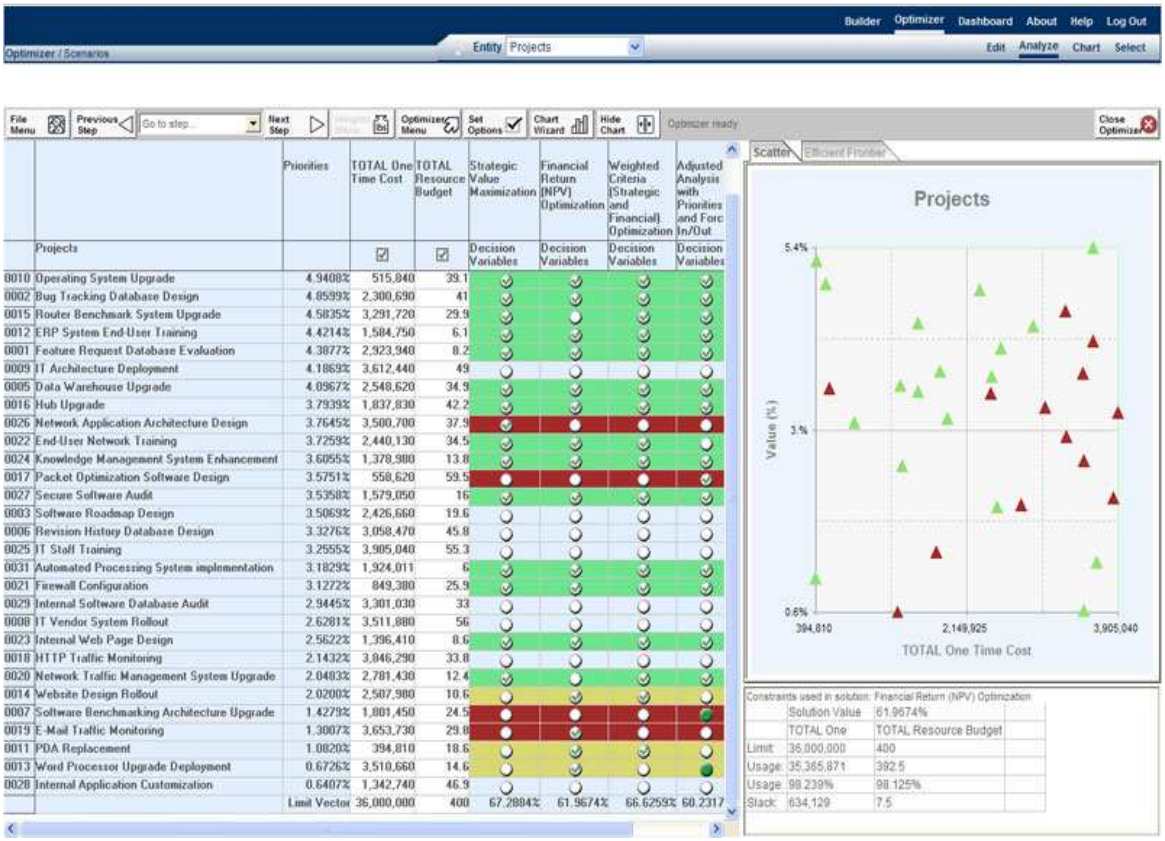


Ilustración 48: Visualización de los estados de los proyectos en EPM Microsoft, extraído de [10]

4.5. Comparativa

4.5.1. Arquitectura y tecnología

En este apartado se hará una comparativa entre las tres herramientas, Project.net, Redmine y Microsoft, desde dos puntos de vista.

El primero de ellos, relacionado con los modos de programación, arquitectura y los costes de licencia.

Aplicación	Lenguaje de programación	Servidor	Cliente	Licencia
Project.net	Java	Apache Tomcat	Navegador	Libre
Redmine	Ruby	WEBrick	Navegador	Libre
Solución EPM Microsoft	.NET	Microsoft Project Server	MS Office Project/Navegador	No libre

De los lenguajes y modelos de programación empleados para desarrollar cada una de las aplicaciones, se puede decir que tanto Ruby como Java, son de uso libre, y .NET requiere licencia. Resulta discutible el optar por una u otra herramienta atendiendo sólo al lenguaje de programación, ya que éste puede considerarse totalmente transparente para una empresa que necesite el uso de una herramienta PPM. La organización compraría el software necesario o lo descargaría, y el hecho de que utilice uno u otro lenguaje de programación debiera ser poco relevante, salvo para aquellas que opten por Redmine o Project.net y quieran añadir nuevas funciones a las aplicaciones, y el hecho de trabajar en Java o Ruby sea considerado por ellas algo importante. Comparando Java con Ruby, se puede decir que Java posee más librerías, lo cual, puede facilitar la creación de nuevos módulos, sin embargo, la facilidad de programación en Ruby es mucho mayor.

En la tabla aparece también el servidor que se instala en una máquina, y al que acceden todos los usuarios, y la aplicación cliente para acceder a ese servidor. En solución de Microsoft se necesitaría una aplicación Microsoft tanto para servidor como para cliente, con su consecuente coste de licencia. En Redmine el servidor que se utiliza es WEBrick, integrado con Ruby on Rails, aunque el servidor Apache también sería completamente compatible con Ruby on Rails, por lo que si WEBrick resultara poco potente se podría optar por utilizar Apache, y como clientes se puede acceder a la plataforma desde cualquier navegador haciendo uso del protocolo HTTP. Esto último es semejante en Project.net, y corre sobre Apache Tomcat. Las ventajas de estas características de Redmine y Project.net sobre Microsoft son claras. Ya se vio en el apartado 3.8.2 la arquitectura completa de la solución EPM de Microsoft. Necesitaría la instalación de varias aplicaciones Microsoft, todas ellas con licencia privada, y como clientes, para funcionar como EPM, sólo podríamos acceder desde un ordenador con Microsoft Office Project, mientras que en los otros dos casos, sólo nos haría falta una conexión a internet.

Por último, desde este primer punto de vista, hay que destacar la ventaja de Redmine y Project.net con respecto a Microsoft al tratarse de herramientas de software libre. Los costes totales de la herramienta pueden considerarse como los costes incurridos al hacerse con ella y su mantenimiento, menos los beneficios aportados tras su utilización. Dentro de los costes de herramienta se incluyen el coste de licencia, cuotas de suscripción, soporte de software, cuotas de mantenimiento, costos relacionados con la personalización del software. Estos costes, cuando se trata de software libre pueden no existir, ya que la licencia es libre y el código fuente de la aplicación es accesible, por lo que, es posible realizar modificaciones a las aplicaciones sin un coste obligatorio, aunque opcionalmente para Project.net existe un servicio de soporte para empresas que es posible su contratación con un determinador coste.

Podría darse el caso de que la organización termine por no usar plenamente la aplicación, que el alcance de su uso no sea el esperado, que tras un cierto tiempo sea abandonado, o que no proporcione información útil, ante esta situación, los riesgos asumidos al optar por Project.net o Redmine también serán menores.

En cuanto a los beneficios de una herramienta PPM dependerán directamente de las funcionalidades que tenga, ya que en función de éstas, resultará más o menos útil en una empresa.

A continuación, se representan las arquitecturas de Redmine y Microsoft EPM. Observamos, como la arquitectura de Redmine es similar a la de Project.net.

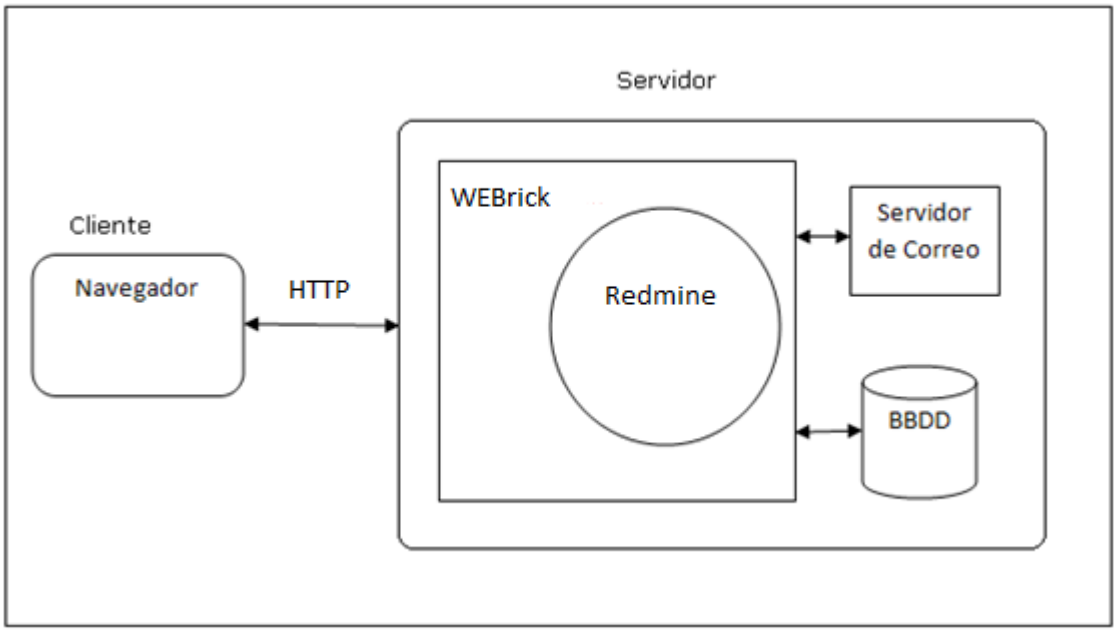


Ilustración 49: Diagrama de bloques de la arquitectura de Redmine

En la figura de la arquitectura de la solución EPM de Microsoft, se ve cómo interactúan cada una de las aplicaciones y herramientas asociadas.

Ilustración 50: Diagrama de bloques de la arquitectura de EPM Microsoft, extraída de [11]

SOAP es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambios de datos XML. Es el protocolo utilizado en la comunicación entre la plataforma Project Web Access y el servidor, y puede ser utilizado también entre el cliente que accede a la plataforma y ésta.
Se muestra a continuación una captura de pantalla de la solución EPM que aparece en la web de Microsoft.

4.5.2. Funcional

Haremos una segunda comparativa entre Project.net, Redmine, y la solución EPM de Microsoft desde el punto de vista de las funcionalidades **[8]**, **[9]** que tienen cada una de ellas.
Se verán características relacionadas con la gestión de proyectos y recursos, y la integración con otras herramientas. En la tabla se dirá cual de las tres aplicaciones las poseen.

Funcionalidad	**Project.net**	**Redmine**	**Microsoft**
Administración de Proyectos			
Guía de proyectos	Sí	Sí	Sí

Plantillas de proyectos	Sí	Sí	Sí
Vistas de Gantt	Sí	Sí	Sí
Subproyectos	Sí	Sí	Sí
Calendario proyectos	Sí	Sí	Sí
Método PERT (riesgos)	No	No	Sí
Controladores de tareas	Sí	Sí	Sí
Planes de actividades	Sí	Sí	Sí
Seguimiento de todos los proyectos en curso de forma centralizada	Sí	Sí	Sí
Trabajo colaborativo	Sí	Sí	Sí
Wiki por Proyecto	Sí	Sí	Sí
Foro por Proyecto	Sí	Sí	Sí
Indicadores predictivos de negocio (costos)	No	No	Sí
Control de finanzas	Sí	No	Sí
Control coste por tarea	No	No	Sí
Administración de recursos			
Recurso con atributos de destreza de varios niveles	Sí	Sí	Sí
Crear equipo	Sí	Sí	Sí
Partes de horas	Sí	Sí	Sí
Integración			
Notificación por correo electrónico	Sí	Sí	Sí
Subida documentos	Sí	Sí	Sí
Posibilidad de cargar y guardar en XML.	Sí	X	Sí
Control Repositorio	No	Sí	No

Observando la tabla, Redmine es la que cuenta con menos funcionalidades relacionadas con la administración de proyectos y el trabajo colaborativo, puesto que no permite controlar el estado de las finanzas.

Cuando se describieron las características de Project.net se habló de que se podía informar al equipo de los costes presupuestados, los estimados y los que llevaban gastados hasta la fecha. Esto no puedo hacerse con Redmine ya que carece de una funcionalidad que le permita controlar el estado financiero del proyecto e informar sobre el mismo.

Salvo esta característica, podemos ver que Project.net y Redmine son herramientas bastante similares en funcionalidades y uso, la mayoría de ellas también están incluidas en la solución de Microsoft. Pero ésta última cuenta con una funcionalidad muy ventajosa, de la cual carecen tanto Project.net como Redmine. En Microsoft, para estimar la duración total de proyecto, es posible, tener en cuenta los riesgos que puedan producirse, haciendo uso del método PERT, ya explicado en el apartado anterior, dando lugar a una fecha estimada de terminación del proyecto más tardía y quizás, más cercana a la realidad.

Otra funcionalidad que posee la solución de Microsoft, y de la cual carecen, tanto Project.net como Redmine, es la de optimización del presupuesto final. En la solución de Microsoft se puede revisar el historial de costes de proyectos anteriores permitiendo estimar mejor los costes para el proyecto actual. La tarea se asigna a unos recursos, que pueden ser tanto a personas, material como equipamiento. Cada uno de esos recursos tiene unas tarifas. A diferencia de Project.net y Redmine, donde cada tarea se asigna sólo a un tipo de recurso, y con recurso se refiere a una persona o grupo de ellas. Esa funcionalidad de la solución de Microsoft permite establecer los costes por cada tarea, y el prepuesto total del proyecto, pudiendo modificar la planificación del proyecto si éste no cumpliera las especificaciones marcadas.

Para concluir este apartado, se destaca la facilidad de implantación en una empresa, tanto de Project.net como Redmine en comparación a la solución EPM de Microsoft, la cual requiere, la instalación de varias aplicaciones de Microsoft, todas ellas con licencia privada. Sin embargo, cuenta con la ventaja de herramientas para el control de costes, modelo de decisiones o predicción de riesgos.