Proyecto Fin de Carrera

Aplicaciones de gestión de proyectos desarrolladas sobre dotProject

Héctor Poveda Sánchez
hpoveda@gmail.com

Facultad de Informática de la Universidad de Murcia
Febrero de 2008

Índice de contenido

Abstract..... 3
Agradecimientos..... 4
1 Resumen y referencias históricas..... 5
1.1 Ventajas de la trazabilidad de requisitos..... 6
1.2 Qué es DotProject..... 9
1.3 La empresa SQA Murcia..... 12
2 Análisis de objetivos y metodología..... 14
2.1 Objetivos estipulados..... 15
2.2 Herramientas utilizadas..... 19
2.3 Recogida de requisitos..... 21
3 Diseño y resolución del trabajo realizado..... 23
3.1 Descripción del sistema actual..... 24
3.2 Análisis de Viabilidad..... 32
3.3 Módulos desarrollados..... 33
Módulo Costs..... 34
Módulo Wages..... 42
Módulo Requisites..... 49
3.4 Otras ampliaciones y mejoras..... 74
4 Conclusiones..... 75
Bibliografía..... 76
Anexos..... 78
I. Integración de dotProject y RequisitePro..... 79
II. Metodología de trabajo en Versas..... 81
III. Plantilla de Acta de Reunión..... 86
IV. Implantación de dotProject en SQA..... 89
V. Otros módulos para dotProject..... 91
VI. Tutorial “proyectos SQA para dotProject”..... 94
VII. Manual de programación en dotProject..... 108

Abstract

En la actualidad, llevar una gestión de proyectos eficiente es vital para conseguir que los proyectos de una empresa culminen dentro del tiempo y presupuesto definidos, especialmente si esta empresa tiene una gran carga de trabajo. Sin embargo, aunque este aspecto debería recibir una atención especial por su parte la realidad es que estos sistemas habitualmente se limitan a planes de trabajo realizados sobre hojas de cálculo, si bien existe cierta tendencia hacia la implantación de software específico normalmente integrado a paquetes como Office de Microsoft.

Por otro lado cada vez son más las empresas que confían en los productos de software libre en su día a día, promoviendo la difusión y mejora de los mismos. Dentro de su oferta de productos destinados a la gestión de proyectos se encuentra dotProject, que es el empleado en este documento.

DotProject fue la opción que mejor se ajustaba a los requisitos de la empresa en la que se ha realizado este Proyecto Fin de Carrera ya que se trata de una plataforma abierta con grandes posibilidades de ampliación. Esta es una empresa dedicada al desarrollo de software que colaboró ofreciendo los medios técnicos y humanos necesarios para llevar a cabo una adaptación a la medida de sus necesidades.

Este trabajo recoge esta experiencia de desarrollo en dicha empresa, estudiando las posibilidades de la plataforma, capturando y analizando los requisitos solicitados por ella y desarrollando varios componentes para la misma. El resultado un software solicitado por la empresa junto a unos manuales de desarrollo, dejando abierta su posible extensión.

Agradecimientos

Me gustaría expresar mi agradecimiento a las personas que han colaborado desde la empresa SQA Murcia para que este proyecto saliera adelante. En especial debo agradecer su colaboración a José María García, Director General de SQA, por facilitarme una definición de requisitos muy completa para el nuevo sistema y gran parte de su tiempo. Ha sido igualmente útil el tiempo y colaboración prestada por Antonio Gallegos, director del área de Gobierno, y Nicolás López, Analista del área de Versas, quienes me han ayudado a conocer las necesidades más importantes de cada departamento.

También quiero agradecer su colaboración por parte de la Universidad de Murcia a Ambrosio Toval por el tiempo y esfuerzo dedicado a revisiones y mejoras de este documento y a Joaquín Lasheras por la información facilitada para la integración de dotProject con RequisitePro.

1 Resumen y referencias históricas

Una gestión de proyectos adecuada es de suma importancia para cualquier empresa, incluso para aquellas con una carga de trabajo no muy elevada. Sin embargo el uso de herramientas de gestión de proyectos está muy poco extendido y, al contrario de lo que se podría pensar, las empresas de desarrollo informático no son una excepción. El uso de técnicas organizativas obsoletas así como la habitual *resistencia al cambio* son sus principales causas.

Afortunadamente cada vez son más las empresas que perciben la informatización de sus procesos como un medio para optimizar tiempo y recursos para destacar entre la competencia. De hecho, tratando este tema con los responsables de cualquier empresa de cualquier ámbito es posible conocer historias sobre iniciativas en este campo que se han intentado llevar a cabo con mayor o menor éxito.

Este proyecto partió de la intención de la empresa SQA Murcia de llevar un seguimiento más exhaustivo de los requisitos de sus proyectos. En el anexo *Metodología de trabajo en Versas* se realiza una descripción de cómo se realizaba este proceso en uno de sus departamentos. En él se observa que, si bien existe una formalización de los requisitos de cada proyecto previa al desarrollo, el fin principal de los mismos es realizar un presupuesto con validez legal de cara al cliente, aunque en la práctica pudieran servir de apoyo al proceso de análisis.

Dado que los requisitos no sólo son un recurso para una adecuada facturación del proyecto sino que también son los *pilares* sobre los que se construirá cada nuevo desarrollo, sería deseable que el sistema de gestión de proyectos llevara un seguimiento de los mismos desde su recogida hasta la implementación de los artefactos relacionados con cada uno de ellos, teniendo en cuenta todas las modificaciones que pudieran sufrir durante su ciclo de vida.

La base de nuestro trabajo será pues la trazabilidad de requisitos.

1.1 Ventajas de la trazabilidad de requisitos

Vamos a comenzar este apartado comentado unas líneas sobre la importancia que tiene la especificación de requisitos del sistema dentro del ciclo de vida del software. En [Glass 2002] se muestran algunos de los *55 hechos* que justifican un seguimiento adecuado de requisitos de los que hemos seleccionado los siguientes:

- *Una de las causas más comunes de proyectos descontrolados es la inestabilidad de los requisitos.*
- *Los errores relacionados con los requisitos son los más caros de corregir durante la construcción del software.*
- *El problema más difícil de corregir, relacionado con los requisitos, es que no sean descubiertos a tiempo requisitos que son relevantes para el proyecto.*
- *Al pasar de los requisitos del problema a los requisitos del diseño, se produce una explosión de "requisitos derivados" (los requisitos que surgen para satisfacer una determinada solución de diseño) causada por la complejidad del proceso de la solución.*

Por su parte en [Blackburn 1996] se menciona:

- *"...con respecto al tiempo utilizado en las diversas etapas del desarrollo de software, la única etapa con una correlación positiva es la del tiempo empleado en la determinación de los requisitos de los clientes: las empresas más productivas dedicaban significativamente más tiempo a estas tareas"*

- *"...el tiempo y el esfuerzo dedicado al prototipado y a otras técnicas para refinar los requisitos de los clientes son amortizadas rápidamente en un ciclo de desarrollo más corto."*

Una vez ha quedado clara la importancia de la fase de recogida de requisitos dentro del ciclo de vida, vamos a comentar las ventajas de establecer un sistema de trazabilidad sobre ellos. En el documento [Letelier 2008] se dan las siguientes razones:

- *Cambios en los requisitos*. Si bien es deseable que los requisitos estén congelados durante todo el ciclo de vida del software, la experiencia dice que hacerlo es inviable. Se producirán cambios en cualquier etapa, y necesitaremos a qué elementos afecta, tanto de la etapa de recogida como de etapas posteriores.

- *Gestión de requisitos*, una de las exigencias del segundo nivel del modelo de calidad CMMi (Repeteable level).

- *Validar la funcionalidad del sistema*. Una trazabilidad desde el requisito hasta el componente software facilitaría la localización de cada especificación a partir del comportamiento del sistema una vez implantado.

- *Mejorar la comunicación con cada participante*. Esto es así siempre que podamos vincular el requisito a las personas interesadas en él, ya sea como cliente, analista, jefe de proyecto, programador, etc.

- *Mejorar la comprensión del sistema*. El hecho de poder acceder a la especificación original de cada elemento del proceso de desarrollo supone una ayuda complementaria para su comprensión.

- *Mejorar la gestión de cambios*, ya que sería posible conocer el alcance que tendría cada modificación sobre los demás componentes del desarrollo. Esto nos daría una idea de su viabilidad, así como de la carga de trabajo que implicaría.

En el documento [PSI 2008] se hace un análisis sobre la complejidad que implica un sistema de trazabilidad de requisitos complejo. De él extraeremos y ampliaremos alguna de sus conclusiones:

- Si hacemos una distinción entre los requisitos de usuario y los de sistema, existirá una relación *uno a pocos* desde los de usuario hacia los de sistema, pudiendo existir requisitos de sistema que no estén ligados a ninguno de usuario.

- La validación del sistema a partir de los requisitos requiere que se lleve la trazabilidad desde su recogida hasta su implementación, lo que implica que exista una cadena de enlaces entre cada requisito y las funciones específicas en el lenguaje de programación empleado que lo satisfagan. Si se trata de programación orientada a objetos el problema se agrava al ser su sintaxis más compleja que en lenguajes procedurales.

- Se espera que la transición entre análisis y diseño sea muy compleja, así como los vínculos entre ambas etapas. Esta complejidad repercute así mismo en la complejidad de un eventual sistema de control de versiones.

- Es importante evitar que la complejidad repercuta en la dificultad de uso del sistema de recogida de requisitos, ya que puede llevar a los desarrolladores a realizar cambios en el código no registrados en el documento de requisitos. El documento explica el concepto de *matriz de trazabilidad de requisitos*, con vínculos hacia etapas anteriores y posteriores, así como con enlaces entre requisitos.

- La correspondencia entre requisito y función del código fuente es *muchos a muchos*, lo que complica aún más la modificación de cada uno de ellos.

- La trazabilidad de requisitos no funcionales es mucho más compleja porque depende de la colaboración entre varios elementos.

Como vemos, aunque se pueden realizar sistemas de tazabilidad de requisitos de gran complejidad, en este trabajo vamos a mostrar un sistema muy simple y de fácil comprensión por parte del usuario final, lo que creemos que repercutirá en una implantación exitosa[1]. Esta sencillez es posible, entre otros motivos, gracias a que el proceso de desarrollo empleado en la empresa es igualmente sencillo, integrando las etapas de análisis y diseño en una, y orientando toda la carga de trabajo de sus proyectos hacia el concepto de *tarea*. En apartados posteriores se analiza a la empresa en más detalle.

1 Así está sucediendo en el departamento Neweb de SQA. En el momento de entregar este proyecto se estaban realizando pequeñas modificaciones al sistema de gestión de requisitos aquí descrito para implantarlo junto al resto de la plataforma.

1.2 Qué es DotProject

DotProject se define como *una aplicación de gestión de proyectos basada en web diseñada para gestionar proyectos y sus funciones de control*. El entorno está programado en *PHP* y *Javascript* como lenguajes en el lado del servidor y *MySQL* como base de datos, lo que lo convierte en un entorno muy accesible a una gran comunidad de programadores.

En lo que respecta a su licencia de distribución, se engloba dentro de los términos de la *GNU General Public License* de la *Free Software Foudation*. Esto implica que todo el código se puede copiar y distribuir libremente mientras se reconozca la autoría del autor original. DotProject puede ser descargado gratuitamente desde su página oficial:

http://www.dotproject.net/

En ella podremos encontrar tanto el programa en sí:

http://sf.net/projects/dotproject

como otros *addons* desarrollados para él:

http://sf.net/projects/dotmods

En el anexo *Otros Módulos para dotProject* se resumen las características de los *addons* que han sido publicados hasta la fecha. Existen traducciones a más de 30 idiomas, entre los que se encuentran Español, Vasco y Catalán. Se pueden encontrar aquí:

http://www.dotproject.net/index.php?name=CmodsDownload

Esto nos da una idea de la repercusión que tiene dotProject en el mundo.

La última versión del programa es la 2.1, que fue lanzada en febrero de 2007, tres años después de la publicación de la primera versión estable del mismo. Desde entonces han sido muchas las empresas que de alguna forma u otra se han interesado por su uso, tanto empresas lucrativas como no lucrativas, si bien no es posible conocer su número exacto ni su éxito una vez implantado.

Sobre la evolución desde la versión 1.0 a la 2.0 cabe decir que la compatibilidad de módulos entre versiones no está garantizada, ya que en la nueva versión se incluyen características, como la nueva gestión de permisos, que alteran seriamente su integración con el entorno. Existe una sección en el foro de soporte dedicada exclusivamente a solucionar las cuestiones relacionadas con los problemas de compatibilidad.

El programa es lo suficientemente robusto como para su implantación en un entorno real. De hecho, poco después de conocer las posibilidades plataforma SQA decidió llevar el seguimiento de todos los proyectos y las tareas que implica a través de ella. Sin embargo existen algunos fallos de menor importancia que se espera sean corregidos en versiones futuras o por la propia empresa, gracias a las ventajas que reporta su licencia de uso.

El mayor problema a la hora de encontrar documentación sobre cómo programar para dotProject es que esta información se encuentra actualmente distribuida entre dos *wikis*:

http://sites.sakienvirotech.com/dp_docs/tiki/

http://docs.dotproject.net

Toda la documentación se está trasladando a la segunda página paulatinamente, aprovechando para revisar conceptos que quedaron obsoletos tras la publicación de la versión 2.0 en abril de 2006.

A la hora de encontrar soporte nos encontramos con que la documentación disponible es muy escasa desde el momento en que necesitemos información más técnica que los simples manuales de usuario. Las cuestiones en el foro de soporte tardan bastante en ser resueltas y rara vez son atendidas.

La causa más probable de esta situación de falta de soporte puede deberse a la existencia del llamado *Priority Support*, que no es más que un servicio de pago para recibir asistencia técnica personalizada. A través de él, el equipo de dotProject se compromete a solucionar cualquier incidencia personalmente, obteniendo la parte proporcional del dinero recaudado en función del número de incidencias resueltas por cada integrante.

Una vez el pago haya sido aceptado utilizando el sistema *PayPal* se habilita el acceso a un foro exclusivo para usuarios del sistema de pago. Los precios de suscripción varían entre los 20,00$ para 5 días de servicio, hasta 500,00$ para la suscripción anual.

Sin querer realizar una comparativa exhaustiva entre las distintas opciones contempladas, mencionaremos alguno de los motivos por los que se decidió realizar una extensión de dotProject frente a otros sistemas similares, como PHProjekt o su antecesor, brújula:

- Las ventajas de dotProject frente a productos comerciales como MSProject no se limitan al aspecto económico. En dotProject la programación de *addons* es muy sencilla de realizar, ya que existe la posibilidad de tomar todo el código del programa como ejemplo así como un práctico tutorial. Para facilitar aún más la labor hemos redactado un manual que se encuentra anexado a este trabajo de nombre *Manual de Programación en dotProject* junto al tutorial también anexo *Proyectos SQA para dotProject*.

- El hecho de que el programa sea gratuito es ventajoso no sólo desde el punto de vista económico, sino que también como una reducción del riesgo que supone una eventual adaptación, especialmente si se lleva a cabo como acuerdo de prácticas en empresa como es el caso.

- A propósito de los riesgos de implantación, el hecho de que toda la base de datos emplee MySQL asegura tanto la portabilidad del contenido registrado en ella en caso de rechazo del sistema una vez implantado, como una garantía de robustez gracias a las facilidades de copias de seguridad y almacenamiento distribuido y replicado que ofrece este sistema. Este factor de portabilidad es de especial importancia después del fracaso del anterior sistema de gestión de proyectos, *Brújula.*

- El concepto *web-based* se ajusta muy bien a una empresa con varias oficinas en Murcia Capital y clientes localizados fuera de la Región. Además, entre sus empleados se dan eventualmente casos de *teletrabajo* durante jornadas no laborables.

- Si bien el concepto *web-based* también está implantado en el sistema *PHProjekt*, este sistema se descartó por los siguientes motivos:
  - Se trata de un sistema creado originalmente por un grupo de desarrollo alemán, por lo que parte de la documentación y contenidos de su web se encuentran únicamente en su idioma. Esto creó cierta *desconfianza* en la plataforma.
  - Se produjo un pequeño problema de incompatibilidad en el momento de su configuración en determinados equipos tras haber empezado a evaluar dotProject satisfactoriamente, lo que llevó a descartar definitivamente esta vía.

- La GUI de dotProject pareció a la empresa muy sencilla, intuitiva y suficientemente atractiva, en especial en lo relativo a las gráficas de Gantt, que permiten visualizar esquemáticamente y de forma clara la planificación de las tareas de cada proyecto.

Para terminar este apartado comentaremos que el número de noticias publicadas en la página web oficial desciende cada año (apenas 7 noticias en 2007 frente a las cerca de 40 de 2004), lo que puede ser un síntoma de que el programa está entrando en una situación de abandono. De acuerdo a los últimos comentarios del equipo de desarrollo esto se debe a una excesiva carga de trabajo, probablemente debida a su servicio de *Priority Support* más que a mejoras en la propia plataforma. En cualquier caso, esto no será un gran inconveniente para una empresa de desarrollo de aplicaciones informáticas como SQA, que siempre tendrá la posibilidad de realizar cualquier tipo de modificación sobre ella para adaptarla a sus necesidades futuras.

1.3 La empresa SQA Murcia

La empresa SQA Murcia S.L. es una empresa con alrededor de 50 empleados distribuidos entre sus tres oficinas en Murcia Capital. Está dividida en tres departamentos, descritos a continuación:

- Neweb, departamento encargado exclusivamente de proyectos relacionados con el desarrollo web.

- Gobierno, departamento dedicado a proyectos relacionados con la administración pública.

- Versas, departamento especializado en la herramienta homónima destinada a empresas del sector hortofrutícola.

Cada departamento dispone de su propio jefe y grupo de desarrollo, así como una metodología de trabajo propia. La dirección de la empresa ha intentado unificar previamente sus formas de trabajo con escaso éxito, disponiendo incluso de documentos que recogen una unificación de sus procedimientos con vistas a la obtención de un estándar de calidad de ISO.

Entre las normas de dicho procedimiento se encontraba un modelo de plantillas en formato *MS Word* destinadas a integrar en en el mismo documento la información recogida durante la reunión con el cliente, los requisitos identificados de la misma, las tareas a realizar por ambas partes y por último la firma de ambas partes para dar validez legal al documento entre otros datos. Se ha anexado esta plantilla bajo el nombre *Plantilla de Acta de Reunión*. También se ha anexado un documento que recoge un breve resumen sobre la metodología de trabajo en el departamento Versas llamado *Metodología de Trabajo en Versas*, cuya visión general se puede extender al resto de departamentos de la empresa.

Cuando surgió la idea de implantar dotProject, la empresa estaba estudiando otras alternativas a su sistema de gestión de proyectos. El principal candidato después de la experiencia con *Brújula* fue *MS Project*, siendo el que mejor se ajustaba a los requisitos de la empresa gracias a su facilidad de integración con las aplicaciones del paquete *MS Office*. En concreto se estaba trabajando en un sistema de gestión de tareas utilizando en sistema *MS Access* que se integraría convenientemente con un sistema de formularios en *MS Excel* y *MS Word* empleando los *Office Web Components* (OWC). Este modelo no implicaba ningún cambio importante en la metodología de trabajo empleada en la empresa hasta ese momento, por lo que era un serio candidato a ser empleado definitivamente. Además, presentaba la ventaja de la portabilidad,

ya que existen multitud de *API*s para diferentes lenguajes que permiten la importación y exportación de documentos en formato *MS Office*.

Brújula fue un sistema desarrollado desde cero que tenía por objetivo integrar toda la gestión de proyectos desde diferentes perspectivas: dirección de empresa, empleado básico, jefe de proyecto, etc. En él se pretendía llevar un seguimiento económico a lo largo de todo el ciclo de vida del proyecto muy detallado, que permitiría realizar comparaciones entre los costes de desarrollo real y la cantidad presupuestada inicialmente. Esta idea de *autoevaluación* se considera por multitud de estándares de calidad como imprescindible para garantizar la rentabilidad de un proyecto (por ejemplo el *Nivel 3* o *Definido* de *CMMi*), ya que es el principal medio para poder realizar nuevos proyectos con suficiente precisión. Por otro lado, aunque no se especificaba un sistema de trazabilidad entre los componentes en desarrollo, sí que quedaba abierta la posibilidad de definirla en un futuro al tratarse de un sistema desarrollado internamente.

Como ya se ha comentado, este sistema fracasó en el momento de su implantación. Esto fue debido entre otros motivos a que:

- La interfaz de usuario era poco intuitiva. Entre las quejas de los usuarios se encontraba la ausencia de atajos de teclado para la introducción de datos.

- El sistema no era lo suficientemente robusto, presentando serios fallos de programación. Esto pudo deberse a la tarea de programación le fue asignada a una persona sin la formación adecuada, lo que por otra parte nos da una idea de la importancia que se dio al sistema de gestión de proyectos en aquel momento.

- Deducido de lo anterior, la empresa no dedicó la suficiente atención al nuevo sistema (especialmente en cuanto a recursos humanos se refiere), por lo que sus probabilidades de éxito eran reducidas.

Curiosamente, esto no está sucediendo tras el periodo de pruebas de implantación con dotProject. Los usuarios están lo suficientemente satisfechos con el nuevo sistema como para aplicarlo a su día a día, lo que abre la puerta a aplicar las mejores ideas del sistema *Brújula* sobre la nueva plataforma.

2 Análisis de objetivos y metodología

En la actualidad no nos consta que existan sistemas de gestión de proyectos específicos que integren una gestión de requisitos adecuada para una empresa de desarrollo de software. De hecho, como se ha comentado anteriormente, el usar dotProject como herramienta de gestión de requisitos partió de la búsqueda alternativas al sistema de gestión de proyectos *Microsoft Project*, que por aquel entonces tenían previsto implantar. Alternativas como *Rational Requisite* están demasiado orientadas al producto, dejando a un lado la gestión del proyecto y sus recursos.

Dado que el objetivo era implantar un producto inexistente en el mercado, iba a ser necesario un nuevo desarrollo. Sin embargo tras la experiencia del *Proyecto Brújula* la empresa era reticente hacia un nuevo desarrollo integral, por lo que el *descubrimiento* de la plataforma dotProject supuso una alternativa mucho más viable.

En los siguientes apartados se describirán de forma resumida los requerimientos de la empresa en cuanto al sistema de gestión de proyectos esperado y cómo se ajustan a este proyecto, que por otra parte serán descritos en detalle en el apartado referente al diseño de la solución. Para terminar el capítulo se hará una breve descripción de las herramientas utilizadas para el desarrollo del presente trabajo.

2.1 Objetivos estipulados

Como ya hemos comentado, existió un precedente en de la empresa como sistema de gestión de proyectos llamado *Brújula*. Existe una especificación de requisitos de este sistema lo suficientemente completa como para, a través de las oportunas revisiones con su autor, definir nuestros propios documentos de requisitos teniendo en cuenta, entre otros factores, las particularidades de la plataforma dotProject.

De acuerdo a lo mencionado y a las limitaciones de tiempo que implica un proyecto fin de carrera hemos fijado sus objetivos siguiendo de tres vías de trabajo, que son:

- La documentación de las técnicas y capacidades de la plataforma con vistas a su ampliación.

- La descripción de un software que cubra las necesidades actuales y futuras de la empresa en cuanto a gestión de proyectos.

- El desarrollo de un software que sirva de punto de partida a futuros desarrollos sobre la plataforma.

La primera vía está cubierta por el presente proyecto en su integridad junto a sus anexos, especialmente los referentes a manuales de programación de módulos para dotProject. El segundo y tercer objetivos se describen dentro de las diferentes secciones del capítulo titulado *Diseño de la solución y trabajo realizado*, así como en *Vías futuras*.

A continuación comentaremos las funcionalidades que la empresa espera de la herramienta a largo plazo:

- *Gestión de los proyectos de cada departamento ajustándose a las características de cada uno de ellos*. Cada departamento se ajusta unas a determinadas características, especialmente a la forma de gestionar sus proyectos. En concreto, el área de Versas desarrolla la práctica totalidad de sus proyectos sobre una plataforma de desarrollo propio que debe adaptarse a cada cliente, por lo que requiere una gestión de proyectos y requisitos distinta a la llevada por otros departamentos.

- *Gestión y seguimiento de las tareas que conforman su desarrollo durante el ciclo de vida del producto*. Como se detalla en el anexo referente a la metodología de trabajo de la empresa, toda la especificación del trabajo a realizar por cada miembro de la

empresa se detalla utilizando el concepto *tarea*. Este concepto abarca desde la recogida de las conclusiones de una reunión de análisis hasta la modificación de un determinado componente software.

- *Gestión y trazabilidad de los requisitos de cada proyecto en particular*, nuevamente teniendo en cuenta las peculiaridades de cada departamento. Aunque en la introducción se han mencionado distintas razones por la que utilizar un sistema de trazas para requisitos, únicamente vamos a enfocar esta característica hacia:

    - *La recogida de requisitos a través de reuniones de análisis*. Tras la entrevista con el ciente, los analistas especificarán formalmente en el sistema las cuestiones recogidas durante dicha entrevista. Esta formalización consistirá básicamente en una especificación de requisitos del sistema, así como en una planificación de la carga de trabajo que implicará su implantación.

    - *La consulta de requisitos asociados a cada tarea*. En concreto, los requisitos podrán ser utilizados como mecanismos de validación de cada componente software durante su implementación, al existir un vínculo transitivo entre el requisito y las tareas que precisa su implantación. Por otro lado, esta facilidad de consulta supone una ayuda a la comprensión de cada tarea.

    - *Permitir la presupuestación por requisito*. Determinados clientes solicitan cambios más o menos complejos sobre un sistema ya implantado que suponen una ampliación de la funcionalidad solicitada inicialmente. Estos cambios requieren un nuevo presupuesto que debe quedar registrado.

    - *Refinamiento de requisitos tras sucesivas reuniones de análisis*. Esta característica supone una de las futuras ampliaciones a partir del sistema entregado. Permitirá acceder a todas las reuniones de análisis que han alterado un requisito, lo que podría ayudar a realizar una definición mejor enfocada hacia propósito real del cliente.

- *Control de los recursos asociados a cada proyecto*, lo que incluye un seguimiento exhaustivo de las tareas y del personal encargado de llevarlas a cabo. Como ya se ha mencionado, esto persigue una mejora de las estimaciones vinculadas a la presupuestación de proyectos y requisitos, si bien su aplicación inmediata es la de realizar una gestión precisa de los recursos a lo largo del tiempo.

Alguno de estos aspectos ya están cubiertos parcialmente por la distribución estándar de dotProject, aunque ninguno se ajusta completamente a los requerimientos de la empresa. A

continuación repasaremos cada uno de los puntos anteriores de acuerdo a la forma en la que dotProject los cubre.

- *Gestión de proyectos.* Como dotProject es un sistema destinado a la gestión de proyectos, este aspecto está muy bien atendido, especialmente en lo que a asignación y planificación de tareas se refiere. Sin embargo no permite filtrar proyectos por departamento ni organizarlos de acuerdo a sus propios criterios. Además, el control de costes que utiliza se reduce a dos campos (*presupuesto tentativo* y *presupuesto real*).

- *Gestión de tareas.* Como acabamos de comentar, está muy bien cubierto por el entorno inicial, permitiendo una práctica descomposición en subtareas. Aunque existe un modo de conocer el coste económico que supone el cumplimiento de una tarea en función del perfil desempeñado en ese momento, no se tiene en cuenta el coste que supone a la empresa las horas *de salario* que ha empleado.

- *Gestión de requisitos.* Esta cuestión no está contemplada en el entorno estándar, por lo que habrá de ser desarrollada en su totalidad, si bien inicialmente pueden expresarse dentro de la descripción de cada tarea o asociándoles ficheros que los contengan.

- *Gestión de recursos.* DotProject incluye un sistema de seguimiento de las horas de trabajo de cada empleado a partir del que se puede llevar un registro sobre el coste real de cada proyecto. Sin embargo esta funcionalidad no es lo suficientemente completa como ya se ha mencionado.

A la vista de lo anterior, el trabajo de ampliación a realizar consistirá en la definición precisa e implementación de nuevos módulos desarrollados a modo de *addons*, de modo que no interfieran ante una eventual actualización de la plataforma. Se ha dejado la puerta abierta al desarrollo de nuevos módulos a través de la información contenida en los manuales anexos de modo que una vez definidos sus requisitos su implementación debería ser relativamente sencilla.

Los módulos desarrollados se describen en profundidad en la sección correspondiente a *Diseño de la solución y trabajo realizado*. Como adelanto describiremos brevemente cada uno de ellos:

- Costs (costes), permite asociar costes a proyectos o tareas determinadas. Este módulo servirá en el futuro para llevar un seguimiento exhaustivo del coste real de cada proyecto.

- Wages (salarios por horas), permite establecer el coste real por horas de trabajo de cada empleado. Este coste real es una estimación del coste que supone a la empresa

cada hora de trabajo del empleado (incluiría el coste salarial, seguridad social y otros), lo que también debe ser tenido en cuenta a la hora de realizar un presupuesto.

- Requisites (requisitos). Este módulo permitirá registrar y visualizar requisitos apropiadamente, así como vincularlos a tareas y proyectos como se explicará más adelante. La definición del *objeto requisito* ha sido el resultado de distintas reuniones entre los analistas de la empresa.

Para promover la filosofía del software abierto y hacer estos módulos accesibles a cualquier desarrollador tanto nacional como internacional todos sus componentes han sido expresados en Inglés (desde el nombre del modulo hasta los comentarios en el código fuente). Por otro lado, se facilita la correspondiente traducción de la interfaz de usuario al Español por lo que el hecho de estar programado en inglés es transparente al usuario final. Todo esto ha sido posible gracias al práctico sistema de traducciones de dotProject.

2.2 Herramientas utilizadas

El software empleado para la codificación de los módulos se ha limitado al uso del editor de texto *Notepad++ v4.5*, debido a que:

- La programación ha sido realizada en su práctica totalidad en los lenguajes interpretados PHP y Javascript, por lo que no ha sido necesario ningún tipo de compilador.

- Este editor dispone de un mecanismo de *sintaxis coloreada* y de *envoltura de sintaxis* en función del lenguaje utilizado, lo que ha resultado ser extremadamente útil en la programación de ficheros de código que llegan a combinar hasta tres lenguajes simultáneamente, como son PHP, HTML y Javascript.

- También dispone de un sencillo sistema de búsqueda y renombrado de variables muy práctico a la hora de adaptar el código de otros módulos a nuestros propósitos.

Los pasos seguidos para implementar cada funcionalidad recogida en el documento de requisitos del módulo fueron los siguientes:

1. Buscar una funcionalidad similar dentro de la interfaz gráfica de usuario (*GUI*) del entorno estándar de dotProject.

2. Dentro del paquete al que pertenece la funcionalidad, encontrar el fichero que la contiene. Para ello seguimos unos sencillos criterios basados en el convenio de nombrado de ficheros que emplea la mayoría de módulos de dotProject.

3. Identificar la porción de código que hace referencia a dicha funcionalidad, normalmente a partir de los textos mostrados por pantalla. Como en la *GUI* estos textos están traducidos, previamente es necesario desactivar el sistema de traducciones de dotProject para encontrar la cadena de texto exacta que aparecerá en el código fuente.

4. Analizar el código encontrado. En el caso de ciertas funciones de cierta complejidad fue necesario acudir a su definición dentro del código interno de dotProject, ya que rara vez estaban documentadas.

5. Por último, adaptar el nuevo código a las exigencias del requisito y validar su funcionalidad.

Por otro lado, en todos los módulos implementados tienen lugar múltiples consultas y transacciones sobre la base de datos, cuya estructura permite incluso acceder a variables del sistema. El sistema de gestión que utiliza es MySQL, por lo que ha sido relativamente sencillo encontrar documentación sobre su funcionamiento.

Para el refinamiento de las operaciones de acceso y modificación de la base de datos hemos utilizado el programa *phpMyAdmin*. Esta aplicación nos permite de una forma sencilla tanto consultar toda la información y estructura de la base de datos, como ejecutar cualquier tipo de sentencia antes de incluirla en nuestro código. Una vez definida la sentencia correcta en este entorno su adaptación al código es inmediata gracias a las instrucciones que provee.

La estructura de la base de datos de dotProject se encuentra disponible a través de este link a la *wiki* antigua:

http://sites.sakienvirotech.com/dp_docs/tiki/tiki-index.php?page=Database+Schema

donde se facilita en formatos *pdf* y *xml*. La versión en *pdf* ha sido exportada de forma inapropiada siendo poco práctica, por lo que hemos empleado la versión en *xml* visualizada con el editor *DBDesigner 4*. Sin embargo el entorno *phpMyAdmin* ha sido lo suficientemente práctico como para no necesitar acudir a este programa con demasiada frecuencia.

El servidor donde hemos instalado dotProject es el conocido como *XAMPP*. *XAMPP* integra en un mismo programa un servidor *Apache*, el sistema de base de datos *MySQL*, intérpretes para *PHP* y *Perl*, así como *OpenSSL* y el ya mencionado *phpMyAdmin*. Esto facilita enormemente los procesos de instalación y configuración, que se pueden realizar de forma muy rápida y sencilla.

Por último, para realizar los diagramas de clases y casos de uso mostrados en este documento se ha empleado la herramienta de libre distribución *ArgoUML*.

2.3 Recogida de requisitos

Para esta etapa del desarrollo se han empleado distintas técnicas de recogida de requisitos que pasamos a comentar brevemente:

- Inicialmente intentamos obtener información a partir del *estudio de la documentación* disponible en la red interna de la empresa. Este proceso sirvió para concluir que la formalización y recogida de documentación para cada desarrollo hasta ese momento no era todo lo completa que cabría esperar, aunque en los últimos años estaba existiendo una mejora al respecto.

- La más habitual fue la *entrevista no estructurada* en lugar de *entrevistas estructuradas* debido a dos motivos principalmente. En primer lugar, todos los participantes tenían conocimientos en el análisis y diseño de sistemas por lo que eran capaces de expresar sus ideas de una forma más formal y coherente de lo que es habitual en un proyecto desarrollado para uno de sus clientes. Y en segundo lugar, la disponibilidad de tiempo de los participantes estaba condicionada a la carga de trabajo que tuvieran en ese instante, ya que las entrevistas tenían lugar durante su horario laboral. Los resultados empleado esta técnica fueron lo suficientemente satisfactorios, y de ellas se obtuvo la mayor cantidad de información.
  Por otro lado comentar que las primeras entrevistas consistían prácticamente en *tormentas de ideas* hasta que paulatinamente se fue definiendo el producto con más detalle.

- Otro elemento básico de recogida de información fue el uso del *prototipado,* técnica que demostró ser muy efectiva. El procedimiento a seguir fue:
  1. Recoger una primera versión de los requisitos a través de una entrevista.
  2. Desarrollar un modelo de pantallas sin ninguna funcionalidad.
  3. Presentar el modelo a los participantes, bien en persona, bien por correo electrónico, para conocer su correctitud.
  4. Actualizar el documento de requisitos e implementarlo.

  Una de las ventajas de las aplicaciones de gestión, como la que nos ocupa, es que el modelo de pantallas se pudo emplear para guiar la implementación de la funcionalidad de cada elemento.

- Otra técnica también utilizada fue la *observación en el puesto de trabajo*, gracias a la que fue posible conocer de primera mano el modo en que las cuestiones planteadas durante las entrevistas debían ser implementadas. Fue especialmente útil para realizar pequeñas correcciones a los requisitos de los módulos.

Otras técnicas como los *Casos de Uso* o el *Joint Application Development* no son aplicables en este entorno debido, como ya se ha comentado, a la experiencia de los participantes. Sin embargo se van a utilizar casos de uso para realizar una descripción del trabajo realizado, ya que su contenido no va dirigido únicamente a los participantes de las entrevistas sino a los desarrolladores que lo continúen.

Por otro lado, el hecho de que dotProject sea un entorno estable facilita la recogida de información, ya que para los usuarios es muy fácil especificar sus necesidades a partir de la GUI existente.

3 Diseño y resolución del trabajo realizado

Comenzaremos este capítulo comentando las características de dotProject en profundidad para pasar a realizar un análisis sobre la viabilidad de la implantación del sistema. El grueso del capítulo consistirá en una especificación detallada de cada módulo implementado.

3.1 Descripción del sistema actual

La distribución estándar de dotProject está formada por los siguientes módulos descritos a continuación:

- **Empresas**, se pueden asociar a proyectos, pudiendo clasificarse como:
  - Cliente
  - Vendedor
  - Proveedor
  - Consultor
  - Gobierno
  - Interno
  - Otros

  Esta clasificación de tipos de empresas puede ser cambiada en cualquier momento por los usuarios con el rol de *administrador del sistema* a través del apartado *Valores del sistema* del módulo *Sistema*.

  Además de los datos típicos de cada empresa, como su información de contacto, nombre, tipo y descripción, cada empresa debe tener un dueño, que se corresponderá con un *usuario.* Curiosamente, dotProject obliga a que el dueño de la empresa tenga su propia cuenta de acceso al sistema, lo que nos hace pensar en un error de diseño. La mejor interpretación que podemos dar a los dueños es considerarlos como a las *personas de nuestra empresa responsables del proyecto para el cliente.*

- **Departamentos**, es una división de una determinada empresa, pudiendo establecerse una relación jerárquica entre ellos. Sólo es útil en empresas con una estructura compleja, como podría ser el Ayuntamiento de Murcia o los propios departamentos de SQA. De los departamentos se registra su nombre, datos de contacto, una descripción opcional y, nuevamente, un propietario, que representa el concepto de *responsable* del departamento.

- **Contactos**, contiene los datos personales de personas con el fin de asociarlos a proyectos y tareas como participantes (*stakeholders*). Una característica interesante es que configurando apropiadamente el servicio de envío de emails es posible enviarles notificaciones ante determinados eventos. Estos participantes no tienen la posibilidad de acceder al sistema como es el caso de los usuarios (descritos en el siguiente punto).

La interfaz de usuario permite mostrar la totalidad de los contactos registrados en el sistema empleando filtros a partir de cadenas de texto y otros atributos.

También se ofrece la opción de exportar el listado de contactos en formato CSV (*comma-separated values*) y la información de cada contacto en formato vCard (pensado para su uso como *electronic business cards*).

- **Usuarios**, son participantes con capacidad para acceder al sistema. Entre sus funciones estarán:
  - Cumplir *tareas,*
  - Gestionar *proyectos,*
  - Administrar el sistema,
  - etc.

  Es decir, serán personas con su cuenta de usuario y contraseña correspondientes. El hecho de ser un usuario implica pertenecer a *contactos* pero no al contrario.

  Para cada usuario se define el nombre de usuario y contraseña (login), nombre y apellidos reales, empresa y departamento al que pertenece, información como *contacto* (que completará a la que aparezca cuando se visualice como un *contacto*) y el tipo de usuario, cuyos valores por defecto son:
  - Usuario por defecto
  - Administrador
  - Director general
  - Director
  - Gerente de sucursal
  - Supervisor
  - Empleado

  pudiendo ser modificados a través de *Valores del sistema* por el administrador.

  A través de la pestaña *Active Sessions* es posible gestionar las conexiones de los usuarios, visualizando su tiempo de conexión y la dirección IP desde la que lo hacen. También es posible desconectarles voluntariamente.

  El modelo de permisos permite configurar tanto los permisos de acceso de cada usuario a cada uno de los módulos del sistema, como a los elementos de dichos módulos. Los permisos de acceso que se pueden especificar para cada módulo o elemento son:
  - Acceso al módulo
  - Vista del módulo
  - Agregar elementos
  - Editar elementos
  - Eliminar elementos

  La asignación de permisos por elementos permite restringir el acceso de cada usuario a determinados proyectos, departamentos o empresas.

Además de lo anterior, cada usuario tiene asignado un rol cuyos valores posibles son configurables a través del módulo de administración del sistema.
A través de la *vista de usuario* el administrador puede conocer en qué proyectos participa cada usuario, ver y modificar los permisos y roles antes descritos y llevar un seguimiento a partir del historial de sus accesos al sistema.
Por último, existen opciones para que cada usuario personalice tanto la GUI como otros tipos de preferencias personales.

- **Proyectos**, módulo en el que se muestran todos los proyectos existentes, pudiendo ser filtrados por propietario, empresa, departamento y estado actual, ya que de otro modo este listado será demasiado extenso.
  Respecto a la entidad *Proyecto,* un proyecto debe tener:
  - Un nombre
  - Un alias o nombre corto
  - Una persona responsable, de tipo usuario, que se interpretará como *jefe de proyecto.*
  - Una empresa (y opcionalmente departamento), que se interpretará como *cliente*.
  - Un valor de prioridad, cuyos valores son editables en *Valores del Sistema.*
  - Un valor de color, para facilitar su visualización en listados extensos.
  - Un tipo de proyecto, cuyos valores también son editables en *Valores del Sistema*. Tiene la misma función que la característica anterior, salvo que en este caso es posible visualizar cada proyecto en una pestaña diferente dentro del módulo.
  - Un valor de estado, con valores editables en *Valores del Sistema*. Este valor se modificará manualmente por el usuario.

  Opcionalmente se puede incluir:
  - Presupuesto tentativo. No se emplea en ningún cálculo interno.
  - Fecha de finalización tentativa. Se calcula automáticamente en función de los valores asignados a las tareas del proyecto.
  - Presupuesto real, para almacenar el coste final del proyecto una vez concluido. No se emplea en ningún cálculo interno.
  - Dos URLs, para añadir información complementaria al proyecto.
  - Una descripción textual del proyecto.

  Se incluye un mecanismo para copiar todas las tareas de otro proyecto existente, lo que da pie a la utilización de *proyectos plantilla.* De hecho, la empresa está empleando estos proyectos plantilla desde la implantación de dotProject, siendo dichas plantillas continuamente revisadas.
  Por otro lado, dotProject facilita la creación de *Informes de Proyecto*. Entre los modelos de informe incluidos en la distribución estándar, se encuentra:
  - *overall.php,* que muestra el total de horas de trabajo de un proyecto.

- *stats.php*, que permite visualizar la evolución de un determinado proyecto a partir del estado de finalización de sus tareas y del tiempo restante para completarlas.
- *taskenddate.php,* para seguir el nivel de cumplimiento de las tareas asignadas a un usuario en un intervalo de tiempo.
- *userperformance.php,* que muestra el rendimiento de cada uno de los trabajadores de un proyecto en un intervalo de fechas determinado.

Todos estos modelos se pueden personalizar a través del código PHP en el que están implementados, por lo que no existe ninguna limitación en este sentido.

- **Tareas**, a cada proyecto se le podrá asignar un conjunto de tareas que podrán ser agrupadas jerárquicamente. Sus atributos principales son:
  - Nombre,
  - Estado, cuyos valores son editables en *Valores del Sistema*.
  - Progreso, para indicar manualmente el grado de consecución de cada tarea en tanto por ciento. Esto es especialmente práctico para llevar a cabo una planificación fiable.
  - Prioridad, cuyos valores son editables en *Valores del Sistema.*
  - Hito, para resaltar la tarea en la gráfica de Gantt de su proyecto.

  Cada tarea dispone de los siguientes atributos agrupados como *detalles*, que permiten asignar:
  - Una descripción de la misión de la tarea, de longitud ilimitada.
  - Un responsable de llevarla a cabo.
  - El tipo de acceso, relacionado con el modelo de permisos de dotProject.
  - Una URL. Esto será de gran utilidad para SQA, que pretende emplear una *wiki* para sus desarrollos.
  - Un tipo de tarea, cuyos valores son editables en *Valores del Sistema.*
  - Tarea padre, para definir una jerarquía de tareas, en la que la tarea padre se descompondría en tareas hijas.
  - Presupuesto, no se emplea en ningún calculo.
  - Contactos, con el mismo significado que en proyectos.
  - Departamento de la empresa para el que se está desarrollando la tarea.

  En otra pestaña se puede realizar una estimación del tiempo que llevará acometer la tarea, especificando la fecha de inicio, de finalización y horas diarias que se dedicarán a ella.

  También es posible gestionar las dependencias entre tareas a través de la pestaña *dependencias.* Esto podrá ser visualizado en Gráficas de Gantt con líneas conectando el final de una tarea con el principio de la siguiente.

  La última pestaña es *recursos humanos,* que permite definir a otras personas *interesadas* en la tarea.

Las tareas asignadas a cada usuario para un día determinado se mostrarán en la primera pantalla que vea al entrar al sistema. Esto es lo que dotProject llama *Vista Diaria*. En ella se muestran las tareas asignadas al usuario que aún no han sido completadas, empleando un color que varía en función del tiempo que resta hasta la finalización estipulada para cada una de ellas.

Cuando un usuario trabaja en una tarea, puede consultar y registrar sus avances en el *historial* de la tarea (log). A través de él se puede especificar:

- El progreso en tanto por ciento.
- Número de horas empleadas, que pueden ser registradas automáticamente mientras el usuario trabaja en ella.
- Código de costo, que se emplea para tarificar de acuerdo al tipo de trabajo o *perfil* que ha realizado el usuario durante ese tiempo. Esto se utiliza para conocer los costes de un proyecto, que además pueden ser distintos dependiendo de la empresa para la que se realice la tarea.
- Descripción del trabajo realizado durante ese tiempo, campo que debe rellenarse obligatoriamente.
- Etc.

- **Calendario**, en el que se muestran todas las tareas del mes para el usuario actual, pudiendo ser filtradas por empresa y evento.

- **Ficheros**, permite disponer de una estructura de directorios para almacenar ficheros que podrán ser asociados a determinados objetos de ciertos módulos. Así mismo, es posible asignar a cada fichero:
  - Un valor numérico referente a la versión de mismo
  - Una categoría, cuyos valores son personalizables a través de *Valores del sistema*
  - Proyecto y tarea a la que estará vinculado
  - Una descripción textual del contenido del fichero

  Además, es posible enviar desde aquí notificaciones a los participantes en el proyecto por correo electrónico.

- **Foros**, se trata de un sistema de foros agrupados por proyecto. Además de los habituales sistemas de filtrado y ordenación de este tipo de servicios, se incluye la opción de exportar los hilos en formato *PDF*.

- **Tickets**, es un sistema de resolución de incidencias basado en el concepto de *ticket*. El procedimiento habitual es el siguiente:
  1. El usuario abre un ticket para indicar su incidencia.
  2. El administrador recibe una notificación para atender la incidencia.
  3. Una vez resuelta la incidencia, el administrador marca la incidencia como resuelta empleando el ticket abierto.
  4. Al usuario se le comunica que su incidencia ha sido resuelta. Si no fuera así o se produjera una incidencia similar, el usuario emplearía el mismo ticket para informar de ella.

- **Sistema**, dispone de las opciones de configuración del sistema, entre las que destacamos:
  - Administrador de traducciones, a través del cual es posible modificar cualquier texto que aparezca en el entorno de dotProject, si bien su intención es permitir su traducción a distintos idiomas.
  - Configuración del sistema, para especificar las preferencias locales y otras.
  - Valores del sistema, a través de los cuales es posible personalizar los estados, tipos de objetos y otros rangos de valores de los objetos del sistema como ya se ha comentado en diversas ocasiones.
  - Editor de campos personalizables, permite, aunque con ciertas restricciones, añadir nuevos campos a los objetos de tipo empresa, tarea, proyecto y calendario, sin alterar la funcionalidad original de dotProject.
  - Gestor de módulos, para instalar/desinstalar, activar/desactivar, visualizar/ocultar módulos de un modo muy sencillo y rápido. Para que un módulo aparezca en el listado bastará con que se incluya un directorio con su nombre en el directorio /dotproject/modules, lo que hace su instalación extremadamente sencilla.

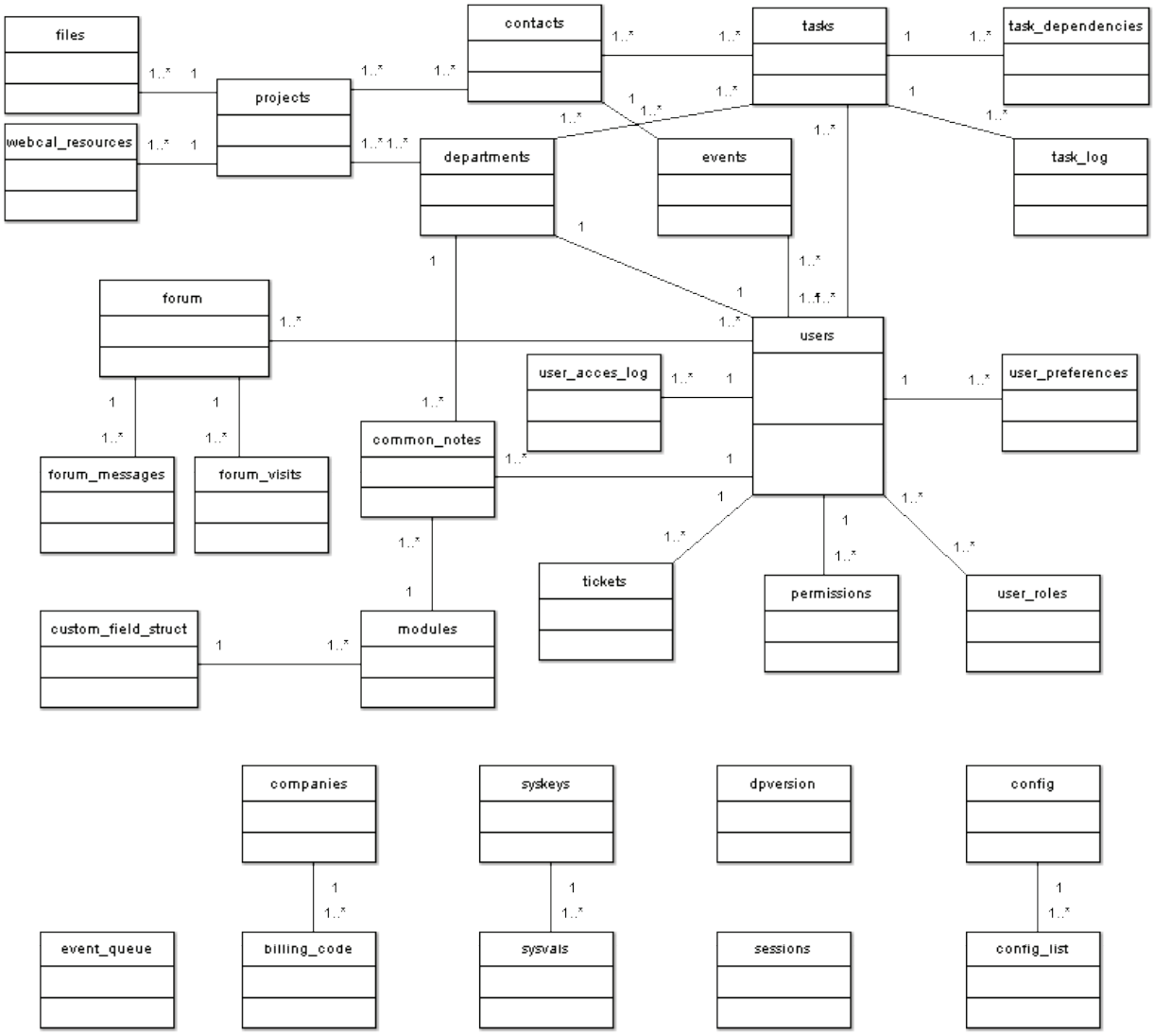
    Al activar un módulo se crean todas las tablas de ese módulo en la base de datos, mientras que al hacerlo visible se hace accesible a todos los usuarios. Por lo tanto no es necesario que un módulo sea visible para poder acceder a sus tablas, pero sí debe estar activo.
  - Roles de usuario, que se emplea para precisar los permisos de acceso de los usuarios a cada módulo concreto.

- **Ayuda**, donde se muestran las vías existentes para obtener ayuda:
  - Documentación *on-line.*
  - Foros de soporte.
  - Sitio de *bugs* y sugerencias.

- Soporte profesional.
- Consultas a su canal *IRC*.

A estos módulos es posible añadir otros desarrollados a modo de addons por otros usuarios disponibles para su descarga en la propia web de dotProject. Para más detalles sobre sus características consúltese el anexo *Otros Módulos para DotProject*.

A continuación se muestra un diagrama de clases obtenido a partir del esquema de la base de datos de dotProject disponible en la web oficial. Es conveniente indicar que dicho esquema de datos es mejorable en muchos aspectos, debido probablemente a problemas de compatibilidad hacia atrás y de reutilización de versiones antiguas.



Como se puede apreciar el esquema es poco claro y nada intuitivo. Por ejemplo, algunos de los errores de diseño de este esquema son:

- La *cardinalidad mínima 1* no tiene sentido en todas las relaciones entre clases.

- Tampoco encontramos sentido a la conexión entre *contacts* y *users* a través de *events,* así como tampoco tiene mucho sentido que *projects* y *tasks* estén conectados a través de *contacts* y *departments*. Hubiera sido más práctico relacionar *tasks* y *projects* directamente.

- *Tasks* y *departments* están conectados directamente, cuando esta conexión no es para nada necesaria, pudiendo ser calculada a través del proyecto al que pertenece la tarea.

- Las dependencias entre tareas se siguen a través de una entidad *task_dependencies*, cuando lo intuitivo hubiera sido una relación de *task* a sobre sí misma.

- Por algún motivo no se ha registrado ninguna relación entre *company* y *project* o *departments,* aunque sí existe una conexión mediante un atributo *project_company* en *project* y un *department_company* en *company.*

- Tampoco comprendemos la relación *uno a uno* entre *users* y *departments*.

Consultando el esquema de base de datos antes mencionado, se puede observar que:

- La asignación de claves secundarias se confunde con las claves foráneas en la mayoría de las entidades. Además estas claves foráneas no se corresponden con las relaciones.

- Parece que los índices se usan para acceder a otras entidades no relacionadas con la entidad en la que se encuentra.

3.2 Análisis de Viabilidad

En este apartado vamos a realizar un breve estudio sobre la viabilidad, tanto de la implantación de dotProject en su distribución estándar, como de las mejoras propuestas en este trabajo. Parte del contenido de este estudio ya se mencionó en el apartado *Qué es dotProject*.

- **Aspectos positivos de la implantación de dotProject**

    1. El entorno dotProject cubre las necesidades básicas de gestión de proyectos de la empresa ya que permite registrar la evolución de los proyectos en el tiempo, así como planificar futuros desarrollos.

    2. El programa es lo suficientemente robusto para su implantación directa.

    3. Tipo de licencia y coste del programa.

    4. Compatibilidad con el sistema de base de datos utilizado (*MySQL*), sencillez y robustez de instalación y configuración.

    5. Filosofía *web-based.*

    6. La gran facilidad de ampliación y adaptación.

    7. La interfaz gráfica de usuario atractiva y práctica.

- **Aspectos negativos de la implantación de dotProject**

    1. Los indicios de abandono de la plataforma.

    2. Problemas a la hora de encontrar soporte técnico.

En conclusión, podemos considerar la implantación del nuevo sistema como **viable**, ya que los factores más importantes para la empresa están cubiertos y los aspectos negativos se pueden salvar por las facilidades que ofrece el tipo de empresa en la que se implantará, como lo demuestra el hecho de que el sistema estaba implantado en el momento de terminar este documento.

3.3 Módulos desarrollados

En las siguientes secciones se va a realizar una descripción detallada de los tres módulos desarrollados. En concreto, para cada uno se mostrará:

- Una explicación textual del módulo, el propósito para el que se ha desarrollado y de qué forma se integra con los demás módulos.
- Un conjunto de casos de uso para facilitar la comprensión de los requisitos del módulo.
- Una especificación de requisitos satisfechos por la versión actual del módulo.
- La integración del módulo desarrollado en el diagrama de clases inicial de dotProject.
- Una descripción textual de algunas propuestas de ampliación y mejora del módulo, que podrá servir como base para su discusión posterior.

Las especificaciones de requisitos se recogen utilizando una plantilla basada en el formato que usa el módulo *Requisites* para registrarlos. La nomenclatura empleada para asignar sus códigos sigue el siguiente esquema:

<módulo>.<ent|vis>.<nombre>

donde:

- módulo expresa el nombre del módulo al que pertenece codificado con tres caracteres
- *ent* es la descripción de una entidad (clase de un objeto)
- *vis* es la descripción de una vista (pantalla).

Dentro de la descripción de cada requisito cada cuestión que debe cumplir aparece numerada empleando la *notación punto*, con lo que todas quedan identificadas intuitiva y unívocamente. Por ejemplo, *cos.ent.coste.1.2* haría referencia a la descripción del atributo *Concepto* de la entidad *coste* dentro del módulo Cost (ver especificación de requisitos más adelante).

Módulo Costs

El primero de los módulos que se van a describir es el de costes de proyecto. Además de los costes asociados a las horas de trabajo del personal, un proyecto o tarea puede requerir otros costes que no pueden ser recogidos por el entorno original dotProject. En concreto intentaremos detallar aún más el mecanismo de seguimiento de costes que los proyectos y tareas ya tienen, que básicamente consisten en *campos presupuesto* para la generación de informes (*reports*) de forma manual.

Por tanto, se desarrollará un módulo de costes (*Costs*) que, asociado a proyectos o a tareas concretas, permita asignarles múltiples objetos con los atributos:

- Concepto
- Valor

con vistas a crear en el futuro un sistema de seguimiento de costes más detallado e integrado en la interfaz estándar de dotProject en lugar de los informes antes mencionados. Esto permitiría tanto calcular el coste de desarrollo real del proyecto como realizar estimaciones sobre él.

Como ejemplo de coste asociado a una tarea podríamos pensar en las *dietas por desplazamiento* para que un analista visite a un cliente situado lejos de Murcia Capital.

Como ejemplo de coste asociado a un proyecto incluiríamos el hardware o licencias software que se incluyen dentro del presupuesto de un proyecto.

- **Casos de uso principales**

Nombre:	**Añadir coste a proyecto**
Descripción:	El usuario añade un nuevo gasto requerido durante el desarrollo de un proyecto previamente presupuestado.
Actores:	Analista
Asunciones:	El proyecto está registrado en el sistema. El usuario tiene permisos de acceso al proyecto y al módulo de costes.
Pasos:	1. El usuario accede al proyecto al que va a añadir el gasto 2. El usuario accede a la pestaña de costes 3. El usuario accede a la vista de creación de costes a través de un botón en esa pestaña 4. El usuario introduce el importe del coste y una descripción del mismo. 4.a. El usuario introduce una tarea a la que asociar el coste, con lo que el coste se asigna a la tarea. 4.b. El usuario no introduce una tarea a la que asociar el coste, con lo que el coste se asigna al proyecto. 5. El usuario pulsa el botón “enviar”. 6. El coste queda registrado en la base de datos asociado al proyecto y/o tarea.
Variaciones:	4. El usuario cancela la creación del nuevo coste, volviendo a la pantalla anterior. 5. El usuario ha introducido un valor no válido en alguno de los campos. Se muestra el aviso correspondiente y se impide su creación.
No funcional:	
Cuestiones:	

Nombre:	**Añadir coste a tarea**
Descripción:	El usuario añade un nuevo gasto requerido a una tarea ya existente.
Actores:	Analista
Asunciones:	La tarea está registrada en el sistema. El usuario tiene permisos de acceso a la tarea y al módulo de costes.
Pasos:	1. El usuario accede a la tarea a la que va a añadir el gasto 2. El usuario accede a la pestaña de costes 3. El usuario accede a la vista de creación de costes a través de un botón en esa pestaña 4. El usuario introduce el importe del coste y una descripción del mismo. 5. El usuario pulsa el botón “enviar”. 6. El coste queda registrado en la base de datos asociado a la tarea.
Variaciones:	4. El usuario cancela la creación del nuevo coste, volviendo a la pantalla anterior. 5. El usuario ha introducido un valor no válido en alguno de los campos. Se muestra el aviso correspondiente y se impide su creación.
No funcional:	
Cuestiones:	

- **Requisitos implementados**

Código: **cos.ent.cost**	
Nombre corto: Descripción de la entidad Cost	
Proyecto: Proyecto Fin de Carrera	Estado: Finalizado

Descripción:

1 La entidad coste tendrá los siguientes atributos:

1.1 Identificador único (id), transparente al usuario.

1.2 Concepto (description), descripción textual de longitud ilimitada en la que se describirá el motivo del gasto.

1.3 Cantidad (amount), importe, en formato decimal (10,2). No se permitirán valores no numéricos. Los valores introducidos deberán ser mayores o iguales a 0.01.

1.4 Proyecto (project), campo que definirá el proyecto para el que se ha registrado el coste.

1.5 Tarea (task), campo opcional que definirá la tarea asociada al gasto. Sólo se podrán seleccionar tareas del proyecto seleccionado.

* Todos los campos son obligatorios salvo que se indique lo contrario.

Código: **cos.vis**	
Nombre corto: Vistas que proveerá el módulo Cost	
Proyecto: Proyecto Fin de Carrera	Estado: Finalizado

Descripción:

1 Vista general, con todos los costes registrados en el sistema.

2 Vista de costes en el módulo Projects, donde se podrán visualizar únicamente los costes vinculados al proyecto o tareas del proyecto.

3 Vista de costes en el módulo Tasks, en la que se mostrarán los costes vinculados a la tarea.

4 Vista de edición, donde se permitirá modificar todos los campos de un coste.

5 Vista de creación, donde se asignarán valores a todos los campos de un coste.

Código: **cos.vis.general**	
Nombre corto: Vista general del módulo Cost	
Proyecto: Proyecto Fin de Carrera	Estado: Finalizado

Descripción:

1 Mostrará todos los costes registrados en el sistema en la ventana principal del propio módulo.
2 Para cada coste mostrado se visualizarán los atributos:
  2.1 Concepto.
  2.2 Cantidad, expresada en euros (€).
  2.3 Proyecto, expresando su nombre completo.
  2.4 Tarea, expresando su nombre completo.
3 El listado se podrá ordenar alfabéticamente haciendo click sobre el título de la columna.
4 Para cada entrada se incluirá un botón para editar cada coste mostrado a través de la vista de edición.
5 Incluirá un botón para crear un coste a través de la vista de creación.
6 Incluirá un botón para eliminar cada coste, previo mensaje de confirmación.

Código: **cos.vis.proyecto**	
Nombre corto: Vista de costes en el módulo Projects	
Proyecto: Proyecto Fin de Carrera	Estado: Finalizado

Descripción:

1 Mostrará todos los costes vinculados al proyecto que se esté visualizando en una pestaña situada en la vista de proyecto.
2 Para cada coste mostrado se visualizarán los atributos:
  2.1 Concepto.
  2.2 Cantidad, expresada en euros (€).
  2.3 Tarea, expresando su nombre completo.
3 El listado se podrá ordenar alfabéticamente haciendo click sobre el título de la columna.
4 Para cada entrada se incluirá un botón para editar cada coste mostrado a través de la vista de edición.
5 Incluirá un botón para crear un coste a través de la vista de creación que asignará por defecto el valor del proyecto visualizado.

Código: **cos.vis.tarea**	
Nombre corto: Vista de costes en el módulo Tasks	
Proyecto: Proyecto Fin de Carrera	Estado: Finalizado

Descripción:

1 Mostrará todos los costes vinculados a la tarea que se esté visualizando en una pestaña situada en la vista de tarea.
2 Para cada coste mostrado se visualizarán los atributos:
  2.1 Concepto.
  2.2 Cantidad, expresada en euros (€).
3 El listado se podrá ordenar alfabéticamente haciendo click sobre el título de la columna.
4 Para cada entrada se incluirá un botón para editar cada coste mostrado a través de la vista de edición.
5 Incluirá un botón para crear un coste a través de la vista de creación que asignará por defecto el valor de la tarea visualizada y el valor del proyecto al que pertenezca esa tarea.

Código: **cos.vis.edición**	
Nombre corto: Vista de edición de coste	
Proyecto: Proyecto Fin de Carrera	Estado: Finalizado

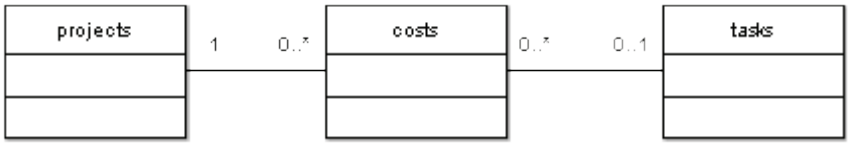
Descripción:

1 Para acceder a esta vista se deberá especificar el identificador del coste a editar.
  1.1 En caso de no especificar ningún valor o que el identificador sea cero se mostrará la vista de creación.
  1.2 En caso de dar un valor no válido se mostrará una pantalla de error.
2 Mostrará todos campos del coste con el identificador especificado excepto el propio identificador.
  2.1 Todos los atributos se podrán editar.
  2.2 El proyecto se seleccionará a través de un menú desplegable con todos los proyectos registrados en el sistema ordenados alfabéticamente.
  2.3 La tarea se seleccionará a través de un menú desplegable con todas las tareas del proyecto seleccionado ordenadas alfabéticamente.
  2.4 Todos los atributos tendrán como valor por defecto su valor anterior.
  2.5 Los atributos sólo podrán tomar valores válidos según el requisito cos.ent.cost.
  2.6 En caso de no asignar ningún valor a un campo obligatorio o de introducir un valor no válido, se mostrará un aviso al usuario y no se permitirá la creación del objeto.
3 Incluirá un botón para cancelar la edición, que volverá a la pantalla anterior.
4 Incluirá un botón para eliminar el coste, previo mensaje de confirmación.

Código: **cos.vis.creación**	
Nombre corto: Vista de creación de costes	
Proyecto: Proyecto Fin de Carrera	Estado: Finalizado
Descripción: 1 Para acceder a esta vista no se deberá especificar ningún identificador de coste o especificar un valor de identificador cero. 2 Se podrán asignar valores a todos campos del salario según el requisito cos.vis.edición. 3 Los valores asignados por defecto serán: 3.1 El identificador, de forma transparente al usuario. 4 Se incluirá un botón para cancelar la creación, que volverá a la pantalla anterior.	

- **Integración con el entorno dotProject**

  Como se puede observar, se reduce a relaciones con los módulos *Projects* y *Tasks*.



Posibles ampliaciones

Como continuación de este trabajo proponemos las siguientes características que podrían ser implementadas en este módulo:

- Añadir un valor de fecha en la que se hace efectivo el gasto. En el caso de compras de hardware o licencias sería útil para distinguir el gasto entre otros de características similares, al igual que en el caso de dietas por desplazamiento permitiría contrastar la información de las nóminas correspondientes.

- Definir un campo de valores predefinidos para el atributo *concepto* junto a un valor de coste *típico* empleando la funcionalidad de *valores del sistema*. Estos valores se establecerían a partir de la experiencia obtenida tras la implantación del módulo.

- Desarrollar un sistema de desglose de costes vinculados a cada proyecto que, además de los costes que recoge actualmente el módulo, muestre también los costes relativos al trabajo del personal y del perfil que hayan desempeñado (analista, programador, etc). En el entorno dotProject ya es posible recoger estos costes relativos al trabajo del personal a través de los *Códigos de coste* que se pueden asignar a los historiales de cada tarea del proyecto, pero esta característica se cubrirá con mayor detalle en el módulo *Wages*, descrito a continuación.
  Dentro de este desglose se podría incluir un valor de desviación entre el *presupuesto tentativo* del proyecto y el coste real que se pudiera emplear a largo plazo para obtener un valor de *calidad de la estimación*, lo que repercutirá en estimaciones más precisas en el futuro.

- En lugar de la opción anterior, emplear la funcionalidad de *Reports* (informes) de dotProject para realizar el desglose de costes vinculados a un proyecto determinado.

Módulo Wages

El historial de las tareas (*log*) permite realizar la tarificación de las horas de trabajo requeridas para cumplir una determinada tarea en función del perfil llevado a cabo por el empleado, pudiendo estas tarifas variar en función de la empresa para la que se esté desarrollando el proyecto. Estos costes son asumidos por el cliente en los proyectos que se tarifiquen por horas.

Aunque la información anterior permite tarificar el proyecto con total precisión de cara al cliente, ese sistema no permite conocer el coste que ha supuesto a la empresa el haber tenido a sus empleados trabajando en esas tareas, ya que un empleado tiene una asignación salarial mensual fija. Estos costes son asumidos por la empresa.

Este segundo punto de vista no está contemplado en dotProject, por lo que será objeto de un nuevo desarrollo. A partir del análisis la información obtenida por ambos tipos de evaluación de costes en proyectos anteriores será posible conocer en el futuro la rentabilidad de proyectos ya terminados. Esto permitirá realizar estimaciones cada vez más precisas en la presupuestación de nuevos desarrollos.

Se desarrollará un módulo de salarios por horas (*Wages*) en el que se podrá asignar un coste/hora de trabajo a cada empleado durante un periodo de tiempo, dejando la evaluación real del coste del proyecto a desarrollos futuros.

- **Casos de uso principales**

Nombre:	**Añadir salario**
Descripción:	El usuario añade un nuevo salario a uno de los empleados de la empresa.
Actores:	Director de área
Asunciones:	El usuario tiene permisos de acceso al módulo de salarios.
Pasos:	1. El usuario accede al módulo de salarios. 2.a. El usuario accede a la vista de creación de salarios a través del botón “nuevo salario” 2.a.1. El usuario selecciona al usuario al que añadir el salario. 2.a.2. El usuario introduce el salario por hora de trabajo. 2.a.3. El usuario introduce el intervalo de fechas de validez del salario, pudiendo no declarar la fecha de fin de validez. 2.a.4. El usuario opcionalmente puede introducir una descripción del salario. 2.b. El usuario accede a la vista de edición de salarios a través del botón “siguiente salario” de alguno de los usuarios registrados. 2.b.1. El usuario introduce el salario por hora de trabajo. El valor anterior aparece como valor por defecto para este campo. 2.b.2. El usuario introduce el intervalo de fechas de validez del salario, pudiendo no declarar la fecha de fin de validez. El valor anterior de fecha fin (si lo hay) más un día aparece como valor por defecto para la fecha inicio. 3. El usuario pulsa el botón “enviar”. 4. El salario queda registrado en la base de datos. 4.a. Si el salario anterior no tiene definida la fecha de fin (es decir, si estaba vigente), esta toma la fecha de inicio del nuevo salario menos un día.
Variaciones:	2. El usuario cancela la creación del nuevo salario, volviendo a la pantalla anterior. 3. El usuario ha introducido un valor no válido en alguno de los campos. Se muestra el aviso correspondiente y se impide su creación.
No funcional:	
Cuestiones:	

- **Requisitos implementados**

Código: **wag.ent.wage**	
Nombre corto: Descripción de la entidad Wage	
Proyecto: Proyecto Fin de Carrera	Estado: Finalizado

Descripción:

1 La entidad salario tendrá los siguientes atributos:

1.1 Identificador único (id), transparente al usuario.

1.2 Usuario (user) al que se aplicará el salario.

1.3 Cantidad (amount), importe, en formato decimal (10,2). No se permitirán valores no numéricos. Los valores introducidos deberán ser mayores que 0.00.

1.4 Fecha inicio (start_date), fecha de inicio de aplicación del salario. No podrá ser posterior a la fecha fin.

1.5 Fecha fin (end_date), campo opcional que representa la fecha de fin de aplicación del salario. No podrá ser anterior a la fecha inicio. Si no toma ningún valor se considerará al salario como vigente.

1.6 Comentarios (comment), campo opcional para dar una descripción del salario, con longitud de texto ilimitada.

* Todos los campos son obligatorios salvo que se indique lo contrario.

Código: **wag.vis**	
Nombre corto: Vistas que proveerá el módulo Wages	
Proyecto: Proyecto Fin de Carrera	Estado: Finalizado

Descripción:

1 Vista general, en la que se visualizarán todos los salarios vigentes y anteriores.

2 Vista de edición, donde se permitirá modificar todos los campos de un salario.

3 Vista de creación, donde se asignarán valores a todos los campos de un salario.

Código: **wag.vis.general**	
Nombre corto: Vista general del módulo Wages	
Proyecto: Proyecto Fin de Carrera	Estado: Finalizado

Descripción:

1 Dispondrá de dos pestañas:
  1.1 “Salarios en vigor”, en la que se mostrará un listado con todos los salarios vigentes.
  1.2 “Todos los salarios”, en la que se mostrarán todos los salarios registrados en el sistema.
2 Para cada salario mostrado se visualizarán los atributos:
  1.1 Nombre del empleado.
  1.2 Apellidos del empleado.
  1.3 Cantidad, expresada en euros (€).
  1.4 Fecha inicio, en formato DD/MM/AAAA.
  1.5 Fecha fin, en formato DD/MM/AAAA. En caso de tratarse de una fecha en vigor este campo no mostrará ninguna fecha.
  1.6 Comentario.
2 Los listados se podrán ordenar alfabéticamente haciendo click sobre el título de la columna.
3 Para cada entrada del listado existirá:
  3.1 Un botón para editar el salario a través de la vista de edición en modo “editar salario”.
  3.2 Un botón para crear el siguiente salario a través de la vista de edición en modo “siguiente salario”.
  3.3 Un botón para eliminar el salario, previo mensaje de confirmación.
4 Incluirá un botón para crear un salario a través de la vista de creación.

Código: **wag.vis.edición**	
Nombre corto: Vista de edición de salario	
Proyecto: Proyecto Fin de Carrera	Estado: Finalizado

Descripción:

1 Para acceder a esta vista se deberá especificar el identificador de salario a editar.
   1.1 En caso de no especificar ningún salario o que el identificador sea cero se mostrará la vista de creación.
   1.2 En caso de dar un valor no válido se mostrará una pantalla de error.
2 Mostrará todos los campos del salario con el identificador especificado excepto el propio identificador.
   2.1 Todos los atributos se podrán editar excepto el campo usuario.
   2.2 Los campos de fechas se editarán a través de sendos “botones calendario”.
   2.3 Los atributos sólo podrán tomar valores válidos según el requisito wag.ent.wages.
   2.4 En caso de no asignar ningún valor a un campo obligatorio o de introducir un valor no válido, se mostrará un aviso al usuario y no se permitirá la creación del objeto.
3 Se podrá acceder a esta vista en dos modos:
   3.1 Modo “editar salario”, que asignará por defecto todos los valores anteriores del salario.
   3.2 Modo “siguiente salario”, que asignará por defecto todos los valores del último salario de ese usuario excepto:
      3.2.1 Su identificador, asignado de forma transparente
      3.2.2 La fecha de inicio, que tomará por defecto el valor de la fecha fin más un día en el caso de que el salario anterior tuviera definida la fecha fin. En otro caso tomará la fecha de inicio del anterior más un día.
      3.2.3 La fecha fin, que no tomará ningún valor por defecto.
      3.2.4 Si el salario anterior estaba vigente, la fecha fin del salario anterior será la fecha fin inicio del nuevo salario menos un día.
4 Se incluirá un botón para cancelar la edición, que volverá a la pantalla anterior.
5 Se incluirá un botón para eliminar el salario, previo mensaje de confirmación.

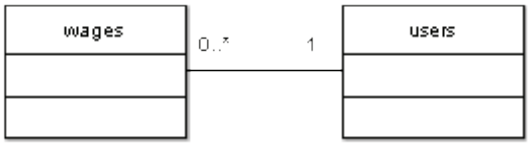
Código: **wag.vis.creación**	
Nombre corto: Vista de creación de salario	
Proyecto: Proyecto Fin de Carrera	Estado: Finalizado

Descripción:

1 Para acceder a esta vista no se deberá especificar ningún identificador de salario o especificar un valor de identificador cero.
2 Se podrán asignar valores a todos campos del salario según el requisito wag.vis.edición.
3 Los valores asignados por defecto serán:
  3.1 El identificador, de forma transparente al usuario.
  3.2 La fecha inicio tomará por defecto el “hoy”.
4 Se incluirá un botón para cancelar la creación, que volverá a la pantalla anterior.

- **Integración con el entorno dotProject**

  En este caso la única conexión que existe en la versión desarrollada es hacia el módulo *Users*.



Posibles ampliaciones

Como continuación de este trabajo proponemos las siguientes características que podrían ser implementadas en este módulo:

- No permitir la introducción de intervalos de fechas que se superpongan en la vista de edición de salario.

- Calcular para los proyectos un valor de rentabilidad, obtenido en función de:
  - El coste (de cara al cliente) de las horas de trabajo invertidas empleando los códigos de costo en función del perfil empleado en sus tareas.
  - El coste real (de cara a la empresa) de las horas de trabajo invertidas por cada empleado en función de su salario.

- Emplear el parámetro anterior para realizar un análisis las presupuestaciones de proyectos anteriores con vistas a mejorarlas en desarrollos futuros.

- Durante la implantación de dotProject en la empresa, el campo descripción del salario era un texto tipo *programador C 2008* (salario de un programador de tipo C durante el año 2008). Esto invita a que este campo pueda ser descompuesto en dos:
  - Categoría salarial
  - Año de aplicación

  con lo que se ampliarían las capacidades de búsqueda de salarios.

Módulo Requisites

A nivel de gestión de proyectos, el ciclo de vida propuesto por la empresa consistirá en:

1 Creación de una primera tarea (que normalmente será una reunión de análisis), de la cual se obtendrán varios requisitos a cumplir por el sistema. Por tanto será necesario vincular requisitos a tareas.

2 Por cada requisito, se crearán las tareas necesarias para satisfacerlo, pudiendo existir varios requisitos satisfechos en parte por una misma tarea. Esto implica vincular tareas a requisitos.

3 Podrán existir requisitos que no procedan de ninguna tarea al no haber sido tratados explícitamente en ninguna reunión. A estos requisitos se les vinculará directamente al proyecto al que pertenezcan.

4 Cada tarea generada puede estar compuesta por otras subtareas (función que ya cubre dotProject). Una tarea puede ser una nueva tarea *de análisis* que genere otras subtareas o requisitos.

El trabajo a realizar consistirá en la elaboración de un módulo de requisitos que se podrá utilizar para recoger los requisitos de cada proyecto de modo que guíe la creación de nuevas tareas. Con esto conseguimos que:

- Exista un vínculo en cada tarea a los requisitos que la dieron lugar, facilitando su comprensión y posterior verificación.

- Exista un vínculo desde cada requisito a la tarea que lo originó, de modo que sea posible conocer la información de la reunión en la que fue generado.

Además, en el futuro podría ampliarse la funcionalidad de este módulo a la redacción de informes y al empleo de métricas de calidad.

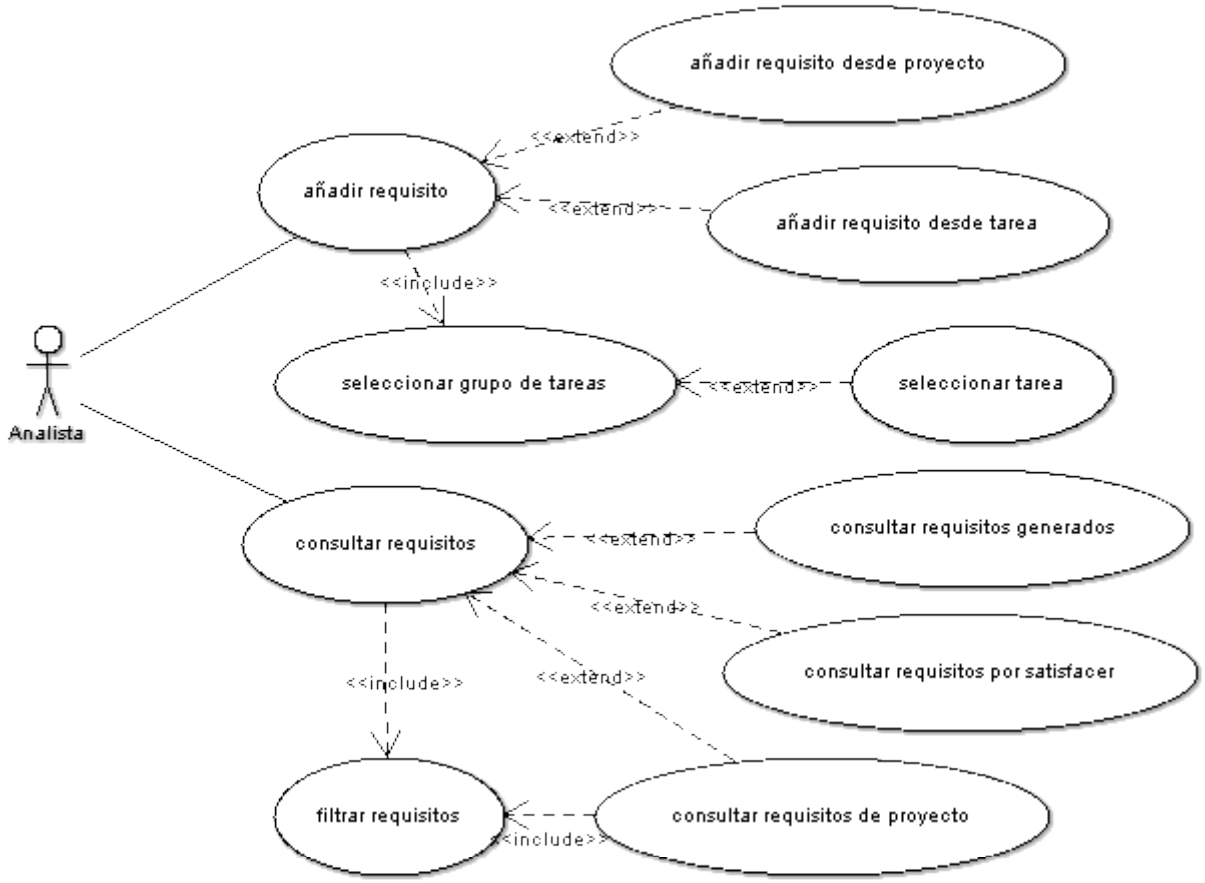
Este módulo estará disponible como un módulo independiente que permita acceder al listado completo de todos los requisitos registrados en el sistema, y como *pestaña* dentro de la descripción de cada proyecto y tarea.

En el caso de la pestaña en proyectos será posible visualizar la totalidad de requisitos del mismo, para lo que será necesario aplicar filtros al listado ya que el número de requisitos de un proyecto será muy elevado.

En el caso de las pestañas en tareas se realizará una distinción entre requisitos generados por esa tarea, en el caso de tratarse de una *tarea de análisis,* y los requisitos que han dado lugar a esa tarea, es decir, los requisitos que esa tarea debe satisfacer.

- **Casos de uso principales**

  Para este módulo se muestra un diagrama de casos de uso, ya que el número de casos de uso descritos es relativamente elevado.



  A continuación se describen todos los casos de uso de este diagrama:

Nombre:	**Añadir requisito**
Descripción:	El usuario añade un nuevo requisito de un proyecto existente.
Actores:	Analista
Asunciones:	El proyecto está registrado en el sistema. El usuario tiene permisos de acceso al proyecto y al módulo de requisitos.
Pasos:	1. El usuario accede al módulo de requisitos. 2. El usuario accede a la vista de creación de requisitos a través del botón “nuevo requisito” 3. El usuario introduce los valores de los campos: o Código del requisito o Nombre corto o Proyecto al que pertenece o Descripción textual detallada o Propietario encargado de modificar su estado o Fecha de recogida o Facturable o Estado del requisito 3.1. El usuario puede introducir opcionalmente los valores de los campos: o Tarea en la que se ha generado, empleando el caso de uso **seleccionar tarea** o Fecha de finalización solicitada por el cliente o Duración presupuestada o URL o Script de base de datos o Tareas que son necesarias para satisfacer el requisito, empleando el caso de uso **seleccionar grupo de tareas** o Clientes a los que se implanta el requisito 3.2. El campo fecha de alta toma el valor “hoy”. 4. El usuario pulsa el botón “enviar”. 5. El requisito queda registrado en la base de datos asociado al proyecto, tareas y clientes.
Variaciones:	3. El usuario cancela la creación del nuevo objeto, volviendo a la pantalla anterior. 4. El usuario ha introducido un valor no válido en alguno de los campos. Se muestra el aviso correspondiente y se impide su creación.
No funcional:	
Cuestiones:	

Nombre:	**Añadir requisito desde proyecto**
Descripción:	El usuario añade un nuevo requisito desde un proyecto existente.
Actores:	Analista
Asunciones:	El proyecto está registrado en el sistema. El usuario tiene permisos de acceso al proyecto y al módulo de requisitos.
Pasos:	1. El usuario accede a la vista de proyecto de uno de los proyectos. 2. El usuario accede a la vista de creación de requisitos a través del botón “nuevo requisito” de la pestaña “requisitos”. 3. El usuario introduce los valores de los campos de igual modo que en caso de uso **Añadir requisito** 3.1. El valor del campo proyecto se asigna automáticamente y no es modificable por el usuario. 4. El usuario pulsa el botón “enviar”. 5. El requisito queda registrado en la base de datos asociado al proyecto, tareas y clientes.
Variaciones:	3. El usuario cancela la creación del nuevo objeto, volviendo a la pantalla anterior. 4. El usuario ha introducido un valor no válido en alguno de los campos. Se muestra el aviso correspondiente y se impide su creación.
Cuestiones:	

Nombre:	**Añadir requisito desde tarea**
Descripción:	El usuario añade un nuevo requisito desde una tarea existente.
Actores:	Analista
Asunciones:	La tarea está registrada en el sistema. El usuario tiene permisos de acceso al proyecto, a la tarea y al módulo de requisitos.
Pasos:	1. El usuario accede a la vista de tareas de una de las tareas. 2. El usuario accede a la vista de creación de requisitos a través del botón “nuevo requisito” de la pestaña “requisitos generados”. 3. El usuario introduce los valores de los campos de igual modo que en caso de uso **Añadir requisito** 3.1. El valor de los campos proyecto y tarea se asigna automáticamente. 4. El usuario pulsa el botón “enviar”. 5. El requisito queda registrado en la base de datos asociado al proyecto, tareas y clientes.
Variaciones:	3. El usuario cancela la creación del nuevo objeto, volviendo a la pantalla anterior. 4. El usuario ha introducido un valor no válido en alguno de los campos. Se muestra el aviso correspondiente y se impide su creación.
Cuestiones:	

Nombre:	**Seleccionar grupo de tareas**
Descripción:	El usuario selecciona un conjunto de tareas de un proyecto concreto
Actores:	Analista
Asunciones:	Se ha seleccionado un proyecto. El usuario tiene permisos de acceso al proyecto, a sus tareas y al módulo de requisitos.
Pasos:	1. Se muestra al usuario la lista de tareas del proyecto seleccionado. 1.a. Todas las tareas seleccionadas antes de iniciar este caso de uso aparecen resaltadas y marcadas. 2.a. El usuario puede marcar cualesquier tareas. 2.b. El usuario puede desmarcar cualesquier tareas. 3. El usuario pulsa el botón “enviar”. 4. Se devuelve el conjunto de tareas marcadas.
Variaciones:	2. El usuario cancela la creación del nuevo objeto, volviendo a la pantalla anterior.
No funcional:	
Cuestiones:	Como el listado de tareas de un proyecto es muy extenso, en futuras versiones deberá ser posible filtrar la lista de tareas del proyecto de acuerdo a algún criterio.

Nombre:	**Seleccionar tarea**
Descripción:	El usuario selecciona una tarea de un proyecto concreto
Actores:	Analista
Asunciones:	Se ha seleccionado un proyecto. El usuario tiene permisos de acceso al proyecto, a sus tareas y al módulo de requisitos.
Pasos:	1. Se muestra al usuario la lista de tareas del proyecto seleccionado. 1.a. Si se había seleccionado una tarea antes de iniciar este caso de uso aparece resaltada. 2. El usuario selecciona una tarea cualquiera. 4. Se devuelve la tarea seleccionada.
Variaciones:	2. El usuario cancela la creación del nuevo objeto, volviendo a la pantalla anterior.
No funcional:	
Cuestiones:	Como el listado de tareas de un proyecto es muy extenso, en futuras versiones deberá ser posible filtrar la lista de tareas del proyecto de acuerdo a algún criterio.

Nombre:	**Consultar requisitos**
Descripción:	Se muestra al usuario una lista de todos los requisitos existentes.
Actores:	Analista
Asunciones:	El usuario tiene permisos de acceso al módulo de requisitos.
Pasos:	1. El usuario accede a la vista general del módulo de requisitos. 2. Se muestra un listado con todos los requisitos del sistema. 2.1. El listado dispone de los siguientes campos: o Código del requisito o Nombre corto, con un enlace a la vista de requisito del requisito correspondiente. o Propietario encargado de modificar su estado o Estado del requisito o Clientes a los que se implanta el requisito o Proyecto al que pertenece o Fecha de recogida o Fecha de finalización solicitada por el cliente o Enlace a la tarea en la que se ha generado, si existe o Enlace a la URL, si existe 2.a. El usuario puede ordenar el listado por campos haciendo click en el título de la columna correspondiente 2.b. El usuario puede filtrar el listado a través del caso de uso **filtrar requisitos**
Variaciones:	
Cuestiones:	

Nombre:	**Consultar requisitos de proyecto**
Descripción:	Se muestra al usuario una lista de todos los requisitos de un proyecto
Actores:	Analista
Asunciones:	El proyecto está registrado en el sistema. El usuario tiene permisos de acceso al módulo de requisitos y al proyecto.
Pasos:	1. El usuario accede a la vista de proyecto de un proyecto concreto. 2. Se muestra un listado de los requisitos de ese proyecto a través de la pestaña “requisitos”. 2.1. El listado dispone de los siguientes campos: o Código del requisito o Nombre corto, con un enlace a la vista de requisito del requisito correspondiente. o Propietario encargado de modificar su estado o Estado del requisito o Fecha de recogida o Fecha de finalización solicitada por el cliente o Enlace a la tarea en la que se ha generado, si existe o Enlace a la URL, si existe 2.a. El usuario puede ordenar el listado por campos haciendo click en el título de la columna correspondiente 2.b. El usuario puede filtrar el listado a través del caso de uso **filtrar requisitos**
Variaciones:	
Cuestiones:	

Nombre:	**Filtrar requisitos**
Descripción:	Permite reducir el número de requisitos mostrados en un listado imponiendo determinadas condiciones en función de los valores de sus campos
Actores:	Analista
Asunciones:	El usuario tiene permisos de acceso al módulo de requisitos.
Pasos:	1. El usuario dispone de un listado de requisitos. 2. El usuario puede asignar valores a los siguientes campos que aplicar como filtro: 2.a. Proyecto, tomando uno de los proyectos registrados en el sistema 2.b. Propietario, tomando a uno de los usuarios registrados en el sistema 2.c. Cliente, tomando a una de las empresas registradas en el sistema 2.d. Estado, tomando uno de los valores posibles para un estado 2.e. Fecha inicio, tomando una fecha 2.f. Fecha fin, tomando una fecha 2.g. Código del requisito, tomando una cadena de texto con comodines. 3. Se actualizará el listado automáticamente restringiendo los elementos mostrados a los que cumplan todas las condiciones solicitadas por el usuario.
Variaciones:	
No funcional:	
Cuestiones:	Puede ser necesario definir un filtro por defecto, ya que los listados de requisitos serán muy extensos.

Nombre:	**Consultar requisitos generados**
Descripción:	Se muestra al usuario una lista de todos los requisitos generados por una tarea
Actores:	Analista
Asunciones:	La tarea está registrada en el sistema. El usuario tiene permisos de acceso a los módulos de requisitos y tareas.
Pasos:	1. El usuario accede a la vista de tarea de una tarea concreta. 2. Se muestra un listado de los requisitos generados por esa tarea a través de la pestaña “requisitos generados”. 2.1. El listado dispone de los siguientes campos: ○ Código del requisito ○ Nombre corto, con un enlace a la vista de requisito del requisito correspondiente. ○ Propietario encargado de modificar su estado ○ Estado del requisito ○ Fecha de recogida ○ Fecha de finalización solicitada por el cliente ○ Enlace a la tarea en la que se ha generado, si existe ○ Enlace a la URL, si existe 2.a. El usuario puede ordenar el listado por campos haciendo click en el título de la columna correspondiente
Variaciones:	
No funcional:	
Cuestiones:	

Nombre:	**Consultar requisitos por satisfacer**
Descripción:	Se muestra al usuario una lista de todos los requisitos que tengan a esa tarea asociada como tarea que es necesario concluir para satisfacer el requisito.
Actores:	Analista
Asunciones:	La tarea está registrada en el sistema. El usuario tiene permisos de acceso a los módulos de requisitos y tareas.
Pasos:	1. El usuario accede a la vista de tarea de una tarea concreta. 2. Se muestra un listado de los requisitos que tengan a esa tarea asociada a través de la pestaña “requisitos por satisfacer”. 2.1. El listado dispone de los siguientes campos: ○ Código del requisito ○ Nombre corto, con un enlace a la vista de requisito del requisito correspondiente. ○ Propietario encargado de modificar su estado ○ Estado del requisito ○ Fecha de recogida ○ Fecha de finalización solicitada por el cliente ○ Enlace a la tarea en la que se ha generado, si existe ○ Enlace a la URL, si existe 2.a. El usuario puede ordenar el listado por campos haciendo click en el título de la columna correspondiente
Variaciones:	
No funcional:	
Cuestiones:	

- **Requisitos implementados**

Código: **req.ent.requisite**	
Nombre corto: Descripción de la entidad Requisites	
Proyecto: Proyecto Fin de Carrera	Estado: Finalizado

Descripción:

1 La entidad requisito tendrá los siguientes atributos:

1.1 Identificador único (id), transparente al usuario.

1.2 Código (code), descripción textual de 1 a 20 caracteres que se utilizará como identificador visible al usuario. Seguirá el convenio de nombres empleado habitualmente por la empresa.

1.3 Nombre corto (name), descripción textual de 1 a 100 caracteres correspondiente al título del requisito.

1.4 Descripción (description), descripción textual de longitud ilimitada en la que se describirá el requisito en detalle.

1.5 Propietario (owner), único usuario responsable de la validación del requisito.

1.6 Estado (status), estado de desarrollo del requisito. Tomará un valor de la variable "RequisiteStatus" definida en "Valores del sistema" para poder ser editados posteriormente. Los valores que esta variable tomará tras la instalación del módulo serán:

0 – Propuesto

1 – Aprobado

2 – En desarrollo

3 – Finalizado

4 – Validado

5 – Facturado

Esta variable será eliminada si el módulo se desinstala.

1.7 Facturable (billable), indica si el requisito es facturable. La facturación se referirá al cliente al cual se implanta el requisito.

1.8 Fecha de recogida (collection date), almacena la fecha en la que se recogió el requisito en el cliente. Nunca será mayor que la fecha de finalización solicitada.

1.9 Fecha de alta (registration date), fecha de creación del requisito. La generará el sistema automáticamente y no podrá ser modificada en el futuro.

1.10 Fecha de finalización solicitada (requested end date), es la fecha propuesta al cliente para la terminación de todas las tareas asociadas al requisito. Nunca será menor que la fecha de recogida.

1.11 Duración presupuestada (budget duration), es la duración en horas presupuestada al cliente para la terminación de todas las tareas asociadas al requisito. Es un campo opcional, ya que los proyectos no tienen por qué

presupuestarse por horas.

1.12 URL, campo opcional que contendrá un vínculo a un recurso asociado al requisito. Podrá tratarse de una dirección web (como la página de una “wiki”) o de cualquier otro documento en línea.

1.13 Script (db script), texto opcional para almacenar un script SQL asociado al requisito en el caso de que éste conlleve una alteración en la base de datos del sistema en desarrollo.

1.14 Proyecto (project), campo que define el proyecto para el que se ha generado el requisito.

1.15 Tarea (task), campo opcional que define la tarea que dio lugar al requisito (que normalmente será una tarea que describa una reunión). Sólo se podrán seleccionar las tareas del proyecto seleccionado.

1.16 Tareas asociadas (associated tasks), lista de tareas creadas a partir del requisito (opcional).

1.17 Lista de clientes (companies list), lista de clientes (companies) afectados con el estado de implantación del requisito (opcional).

* Todos los campos son obligatorios salvo que se indique lo contrario.

Código: **req.vis**	
Nombre corto: Vistas que proveerá el módulo Requisites	
Proyecto: Proyecto Fin de Carrera	Estado: Finalizado

Descripción:

1. Vista general, en la que se podrán visualizar todos los requisitos registrados en el sistema.
2. Vista de requisitos en el módulo Projects, donde se podrán visualizar únicamente los requisitos vinculados al proyecto
3. Vista de requisitos en el módulo Tasks, en la que se podrán visualizar los requisitos vinculados a la tarea.
4. Vista de requisito, donde se visualizarán los campos de un requisito.
5. Vista de edición, donde se permitirá modificar todos los campos de un requisito.
6. Vista de creación, donde se asignarán valores a todos los campos de un requisito.

Código: **req.vis.general**	
Nombre corto: Vista general del módulo Requisites	
Proyecto: Proyecto Fin de Carrera	Estado: Finalizado

Descripción:

1 Dispondrá de dos pestañas, que por defecto se visualizarán en modo “plano”:

1.1 Filtro de búsqueda, que permitirá aplicar un filtro a los requisitos mostrados en la lista de requisitos.

1.1.1 Todos los campos del filtro podrán dejarse en blanco.

1.1.2 Los campos del filtro tomarán por defecto el valor de la búsqueda anterior.

1.1.3 El listado se actualizará tras seleccionar cualquier valor en un menú desplegable, botón calendario o pulsando la tecla intro tras introducir una cadena de búsqueda.

1.1.4 Se podrá filtrar por:

1.1.4.1 Proyecto al que pertenece, con un menú desplegable en el que se mostrarán todos los proyectos registrados en el sistema ordenados alfabéticamente.

1.1.4.2 Propietario, con un menú desplegable en el que se mostrarán los nombres de usuario ordenados alfabéticamente de todos los usuarios registrados en el sistema.

1.1.4.3 Cliente que lo solicita, con un menú desplegable en el que se mostrarán todos los clientes registrados en el sistema ordenados alfabéticamente.

1.1.4.4 Estado del requisito, con un menú desplegable en el que se mostrarán todos los valores de la variable “RequisiteStatus”.

1.1.4.5 Código, con un campo de texto junto a un botón de búsqueda. En el texto de búsqueda podrá tener la forma de “consulta MySQL”, permitiendo el uso de los mismos comodines (%, _, etc).

1.1.4.6 Intervalo de fechas de recogida del requisito, a través de dos botones calendario, pudiendo definir:

- ninguna fecha, con lo que no se aplicará el filtro.
- sólo fecha inicio, mostrando los requisitos con fecha de recogida mayor que la indicada.
- sólo fecha fin, mostrando los requisitos con fecha de finalización solicitada menor que la fecha indicada.
- fecha inicio y fecha fin, mostrando los requisitos que cumplan las dos condiciones anteriores.

Se mostrará un mensaje de error si la fecha fin es anterior a la de inicio.

1.2 Lista de requisitos, que mostrará un listado de todos los requisitos registrados en el sistema, filtrado con las opciones antes mencionadas. Para cada requisito mostrado se visualizarán los atributos:

1.2.1 Código.

1.2.2 Nombre corto, mostrando únicamente los primeros 40 caracteres. Si contiene más, se podrá visualizar el texto completo a través de un tooltip. Haciendo click sobre él se accederá a la vista de requisito.

1.2.3 Propietario, mostrando el nombre de usuario.

1.2.4 Estado, mostrando el valor de “RequisiteStatus” correspondiente.

1.2.5 Cliente, mostrando todos los clientes asociados al requisito.

1.2.6 Proyecto.

1.2.7 Fecha de recogida, en formato DD/MM/AAAA.

1.2.8 Fecha de finalización solicitada, en formato DD/MM/AAAA.

1.2.9 Vínculo a la tarea creadora, a través de un icono. Se abrirá en la misma ventana del navegador.

1.2.10 Vínculo a la URL del requisito si la hay, a través de un icono. Se abrirá en una nueva ventana o pestaña del navegador.

Las filas podrán ser ordenadas alfabéticamente haciendo click en el título de la columna. La columna cliente no será ordenable por ser un campo multivaluado.

2 Existirá un botón para crear un nuevo requisito empleando la vista de creación.

Código: **req.vis.proyecto**	
Nombre corto: Vista de requisitos en el módulo Projects	
Proyecto: Proyecto Fin de Carrera	Estado: Finalizado

Descripción:

1 Mostrará todos los requisitos del proyecto en una pestaña situada en la vista de proyecto.

2 Existirá un filtro de búsqueda que permitirá aplicar un filtro a los requisitos mostrados.

1.1 Todos los campos del filtro podrán dejarse en blanco.

1.2 Los campos del filtro tomarán por defecto el valor de la búsqueda anterior.

1.3 El listado se actualizará tras seleccionar cualquier valor en un menú desplegable, botón calendario o pulsando la tecla intro tras introducir una cadena de búsqueda.

1.4 Se podrá filtrar por:

1.4.1 Propietario, con un menú desplegable en el que se mostrarán los nombres de usuario ordenados alfabéticamente de todos los usuarios registrados en el sistema.

1.4.2 Estado del requisito, con un menú desplegable en el que se mostrarán todos los valores de la variable “RequisiteStatus”.

1.4.3 Código, con un campo de texto junto a un botón de búsqueda. En el texto de búsqueda podrá tener la forma de “consulta MySQL”, permitiendo el uso de los mismos comodines (%, _, etc).

1.4.4 Intervalo de fechas de recogida del requisito, a través de dos botones calendario, pudiendo definir:

- ninguna fecha, con lo que no se aplicará el filtro.
- sólo fecha inicio, mostrando los requisitos con fecha de recogida mayor que la indicada.
- sólo fecha fin, mostrando los requisitos con fecha de finalización solicitada menor que la fecha indicada.
- fecha inicio y fecha fin, mostrando los requisitos que cumplan las dos condiciones anteriores.

Se mostrará un mensaje de error si la fecha fin es anterior a la de inicio.

2 Los atributos mostrados en la lista de requisitos serán:

2.1 Código.

2.2 Nombre corto, mostrando únicamente los primeros 40 caracteres. Si contiene más, se podrá visualizar el texto completo a través de un tooltip. Haciendo click sobre él se accederá a la vista de requisito.

2.3 Propietario, mostrando el nombre de usuario.

2.4 Estado, mostrando el valor de “RequisiteStatus” correspondiente.

2.5 Fecha de recogida, en formato DD/MM/AAAA.

2.6 Fecha de finalización solicitada, en formato DD/MM/AAAA.

2.7 Vínculo a la tarea creadora, a través de un icono. Se abrirá en la misma ventana

del navegador.

2.8 Vínculo a la URL del requisito si la hay, a través de un icono. Se abrirá en una nueva ventana o pestaña del navegador.

Las filas podrán ser ordenadas alfabéticamente haciendo click en el título de la columna.

3 Existirá un botón para crear un nuevo requisito empleando la vista de creación, asignando por defecto el valor del proyecto visualizado.

Código: **req.vis.tarea**	
Nombre corto: Vistas de requisitos en el módulo Tasks	
Proyecto: Proyecto Fin de Carrera	Estado: Finalizado

Descripción:

1 Existirán dos pestañas en este módulo:
  1.1 Requisitos generados por la tarea, que mostrará una lista con todos los requisitos que haya generado la tarea (normalmente será una tarea que describa una reunión de análisis).
    1.1.1 Existirá un botón para crear un nuevo requisito empleando la vista de creación, asignando por defecto el valor de la tarea visualizada y del proyecto al que pertenece.
  1.2 Requisitos a satisfacer por la tarea, que mostrará una lista con todos los requisitos que tengan asociada a esa tarea.
2 En ambos casos se mostrarán los siguientes atributos del requisito:
  1.1 Código.
  1.2 Nombre corto, mostrando únicamente los primeros 40 caracteres. Si contiene más, se podrá visualizar el texto completo a través de un tooltip. Haciendo click sobre él se accederá a la vista de requisito.
  1.3 Propietario, mostrando el nombre de usuario.
  1.4 Estado, mostrando el valor de “RequisiteStatus” correspondiente.
  1.5 Fecha de recogida, en formato DD/MM/AAAA.
  1.6 Fecha de finalización solicitada, en formato DD/MM/AAAA.
  1.7 Vínculo a la tarea creadora, a través de un icono. Se abrirá en la misma ventana del navegador.
  1.8 Vínculo a la URL del requisito si la hay, a través de un icono. Se abrirá en una nueva ventana o pestaña del navegador.

  Las filas podrán ser ordenadas alfabéticamente haciendo click en el título de la columna.

Código: **req.vis.requisito**	
Nombre corto: Vistas de requisito	
Proyecto: Proyecto Fin de Carrera	Estado: Finalizado

Descripción:

1 Permitirá visualizar todos los atributos que se hayan definido de un determinado requisito en modo de sólo lectura.
   1.1 Se podrá hacer click sobre las tareas asociadas al requisito, lo que mostrará la vista de esa tarea en la misma ventana del navegador.
   1.2 Se podrá hacer click sobre los clientes asociados al requisito, lo que mostrará la vista de esa empresa en la misma ventana del navegador.
   1.3 Se podrá hacer click sobre la URL del requisito, lo que abrirá el recurso al que referencia en una nueva ventana o pestaña del navegador. Sólo se mostrarán los últimos 45 caracteres de la URL para facilitar su visualización.
   1.4 Los campos que no tomen ningún valor no se mostrarán.
2 Incluirá botones para:
   2.1 Editar requisito, que llevará a la pantalla de edición del requisito.
   2.2 Mostrar listado completo de requisitos.
   2.3 Volver atrás.
   2.4 Borrar requisito, pidiendo la correspondiente confirmación.

Código: **req.vis.edición**	
Nombre corto: Vista de edición de requisitos	
Proyecto: Proyecto Fin de Carrera	Estado: Finalizado

Descripción:

1 Para acceder a esta vista se deberá especificar el identificador del requisito a editar.

1.1 En caso de no especificar ningún valor o que el identificador sea cero se mostrará la vista de creación.

1.2 En caso de dar un valor no válido se mostrará una pantalla de error.

2 Mostrará todos campos del requisito con el identificador especificado excepto el propio identificador.

2.1 Todos los atributos tendrán como valor por defecto su valor anterior.

2.2 Todos los atributos se podrán editar.

2.2.1 El proyecto se podrá modificar a través de un menú desplegable con todos los proyectos registrados en el sistema ordenados alfabéticamente.

2.2.1.1 Si se modifica el valor de este campo se borrará el contenido de los campos tarea y tareas asociadas para evitar inconsistencias.

2.2.2 La tarea se podrá modificar a través de un botón “buscar tarea” que abrirá una ventana con una lista de todas las tareas existentes para el proyecto seleccionado.

2.2.2.1 El usuario seleccionará una haciendo click sobre ella.

2.2.2.2 La tarea seleccionada anteriormente aparecerá resaltada.

2.2.3 El propietario se podrá editar empleando un menú desplegable en el que se mostrarán todos los nombres de usuario de los usuarios registrados en el sistema en orden alfabético.

2.2.4 La fecha de recogida se podrá editar empleando un “botón calendario”.

2.2.5 La fecha de alta no se podrá modificar.

2.2.6 El campo facturable se podrá modificar empleando un “checkbox”.

2.2.7 El estado se podrá modificar seleccionando con un menú desplegable los valores de la variable “RequisiteStatus”.

2.2.8 La fecha de finalización solicitada se podrá editar empleando un “botón calendario”.

2.2.9 La lista de tareas asociadas se podrá modificar a través de un botón “editar lista” que abrirá una ventana con una lista de todas las tareas existentes para el proyecto seleccionado.

2.2.9.1 Junto a cada tarea se mostrará un “checkbox” para añadirla o eliminarla de la lista.

2.2.9.2 Las tareas seleccionadas anteriormente aparecerán resaltadas.

2.2.9.3 Si no se ha seleccionado ningún proyecto no se abrirá la ventana y se mostrará un aviso al usuario.

2.2.10 La lista de clientes se podrá modificar a través de un botón “editar lista”

que abrirá una ventana con una lista de todos los clientes registrados en el sistema.

2.2.10.1 Junto a cada cliente se mostrará un “checkbox” para añadirlo o eliminarlo de la lista.

2.2.10.2 Los clientes seleccionados anteriormente aparecerán resaltados.

2.3 Los atributos sólo podrán tomar valores válidos según el requisito req.ent.requisite.

2.4 En caso de no asignar ningún valor a un campo obligatorio o de introducir un valor no válido, se mostrará un aviso al usuario y no se permitirá la creación del objeto.

3 Al terminar la edición se volverá a la vista de requisito.

4 Incluirá botones para:

4.1 Cancelar la edición, que volverá a la pantalla anterior.

4.2 Eliminar el requisito, previo mensaje de confirmación.

4.3 Mostrar el listado completo de requisitos.

4.4 Volver atrás.

Código: **req.vis.creación**	
Nombre corto: Vista de creación de requisitos	
Proyecto: Proyecto Fin de Carrera	Estado: Finalizado

Descripción:

1 Para acceder a esta vista no se deberá especificar ningún identificador de coste o especificar un valor identificador cero.

2 Se podrán asignar valores a todos campos del requisito según el requisito req.vis.edición.

3 Los valores asignados por defecto serán:

3.1 Se asignará un nuevo identificador de forma transparente al usuario.

3.2 El propietario tomará como valor por defecto al usuario conectado.

3.3 La fecha de recogida tomará como valor por defecto “hoy”.

3.4 La fecha de alta tomará como valor por defecto “hoy”.

3.5 El estado tomará como valor por defecto al primero de la lista de valores de la variable “RequisiteStatus”.

3.6 Los demás campos no tomarán ningún valor por defecto.

4 Incluirá botones para:

4.1 Cancelar la creación, que volverá a la pantalla anterior.

4.2 Mostrar el listado completo de requisitos.

4.3 Volver atrás.

- **Integración con el entorno dotProject**

  Este módulo está relacionado con los módulos:

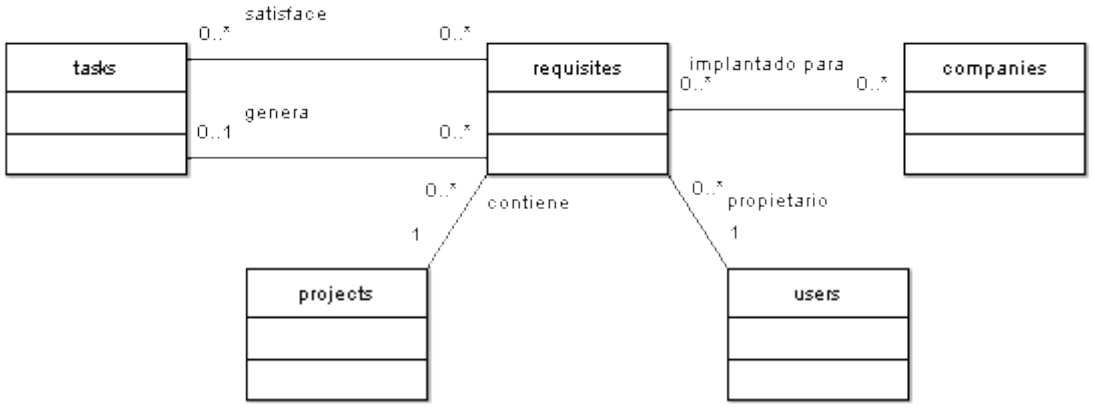
  - *Users*, ya que cada requisito dispone de un responsable (propietario).

  - *Companies*, donde los clientes a los que se implanta el requisito se recogen como compañías.

  - *Projects*, ya que todo requisito pertenece a un proyecto.

  - *Tasks*, cuyas relaciones con el módulo de requisitos existen debido a que un requisito puede ser generado por una tarea y a que un requisito está relacionado con las tareas que se encargan de satisfacerlo.

  A continuación se muestra el diagrama de clases correspondiente:



Posibles ampliaciones

Como continuación de este trabajo proponemos las siguientes características que podrían ser implementadas en este módulo:

- Mejorar el sistema de trazabilidad “hacia atrás” por el que es posible conocer el origen de un requisito. Un requisito puede sufrir varias modificaciones o refinamientos a lo largo del ciclo de vida del sistema. Una forma de implementar esta característica sería la siguiente:

    - No se permitiría la edición directa de un requisito en estado “aprobado” o posterior.

    - Estos requisitos sólo se pueden editar a través de una tarea (que describa por ejemplo una segunda reunión de análisis) a través de un botón ”refinar requisito”.

    - Con este botón se podrá seleccionar uno de los requisitos existentes, pasando acto seguido a su edición. El modo de edición debería estudiarse detalladamente, ya que podría ser interesante conservar los valores anteriores tomados por algunos campos. Supondremos que únicamente se permite modificar la descripción del requisito, perdiéndose el contenido anterior.

    - Una vez refinado el requisito sería posible acceder a través de él tanto a la tarea que lo creó, como a todas las tareas que lo hubieran modificado para identificar las reuniones que han dado lugar a cada requisito.

- Aplicar un filtro por defecto a los requisitos mostrados en las vistas general y de proyecto, ya que estos listados serán muy extensos.

- Del mismo modo, mejorar el sistema de selección de tareas de modo que sea posible filtrar sus listados, dado que el número de tareas será también muy elevado.

- Permitir la creación de tareas desde la vista de creación y edición de requisitos, bien empleando la vista de creación de tareas existente en dotProject o bien empleando una interfaz propia, que además permitiera la creación de determinadas tareas de forma automática.

- Insertar en los requisitos mostrados a en la vista de una tarea un menú desplegable que permita al usuario encargado de cumplirla asignar al requisito los estados “en

desarrollo” o “finalizado”. Posteriormente, el propietario del requisito sería el encargado de asignarle el valor “validado” y/o “facturado”. De este modo se podría seguir fácilmente la evolución del requisito.

- Se podría realizar un sistema de trazabilidad “horizontal” entre requisitos a través de las tareas que tengan asociadas. Por ejemplo, si dos requisitos comparten una misma tarea, se podría considerar que estos requisitos están relacionados. Esta característica debería refinarse a través de la experiencia de uso del sistema.

- También a partir de la experiencia obtenida tras el uso prolongado de este sistema, se podría implementar un sistema para importar los requisitos de un proyecto anterior a modo de “plantilla” para un nuevo desarrollo. En este caso sería conveniente que existiera algún tipo de vínculo entre el requisito original y el requisito adaptado.
  En concreto los requisitos de proyecto (procedimientos, costes, proceso de validación y verificación, etc) no están ligados a ninguna tarea concreta, por lo que son buenos candidatos a formar parte de estos “proyectos plantilla”.

- En este mismo módulo, o bien a través de los *reports* de dotProject, se podría idear un método de generación de documentos para el cliente y para la empresa, empleando tanto los requisitos obtenidos como la información recogida en las tareas *de análisis*.

- Por último, se propone integrar el sistema de trazabilidad de requisitos descrito en este documento con el programa RequisitePro. Dado que ambos entornos emplean internamente sistemas de bases de datos incompatibles no sería posible mantener una base de datos única compartida.
  Esta integración sólo se podría realizar a través de la importación/exportación de datos entre ambas plataformas en situaciones puntuales. En este caso, se proponen las siguientes soluciones:

  - Una posibilidad sería mantener dos copias de los mismos requisitos de un mismo proyecto en ambas plataformas. Para ello se emplearía una función de *actualización* en el momento de abrir el proyecto en alguna de las plataformas que realizaría las modificaciones en los elementos de la base de datos propia para asegurar la consistencia entre ambas.

  - Como probablemente la opción anterior consumiría excesivos recursos y la función principal de dotProject no es la gestión de requisitos, se podría utilizar RequisitePro para recogida y modificación y dotProject sólo para consulta. Para ello sólo serían necesarios mecanismos de importación en dotProject, simplificando la implementación de esta solución. De este modo se dispondría de un entorno más

adecuado para la recogida y de un sistema de trazabilidad hacia atrás eficiente para la planificación de tareas.

- Una última propuesta sería utilizar la base de datos de dotProject para almacenar *plantillas de conjuntos de requisitos*, con vistas a formar parte de plantillas de proyectos. Estas plantillas podrían ser desarrolladas sobre dotProject y refinadas sobre este mismo entorno a partir de la experiencia en los proyectos sobre los que se apliquen.

  La implementación de esta característica se podría realizar a través de la exportación de requisitos de dotProject a RequisitePro en el momento de iniciar un nuevo proyecto. Además, para facilitar el refinamiento de las plantillas de dotProject se podría mantener un vínculo hacia las distintas implantaciones de cada una en RequisitePro.

En el anexo *Integración con RequisitePro* se resumen los mecanismos de integración principales de RequisitePro y dotProject.

3.4 Otras ampliaciones y mejoras

Además de las mejoras propuestas para cada módulo en este mismo capítulo, proponemos las siguientes características que también podrían ser implementadas en un futuro:

- Como ya se comentó en un capítulo anterior, la empresa dispone de tres departamentos diferentes que, por las características del tipo de software que desarrollan, agrupan sus proyectos de forma distinta. En este caso se podría realizar una copia del módulo proyectos para cada departamento para no alterar su funcionalidad original.

- Se podría realizar una modificación del modelo de tareas actual de dotProject, en el que la persona encargada de cumplirla pudiera reajustar la cantidad de tiempo necesaria para cumplirla. Este reajuste iría acompañado de una justificación que podría tomar los siguientes valores:
  - Se realizó una mala planificación o no existe línea de trabajo.
  - Etapa previa incompleta.
  - Mala gestión del cliente, se han propuesto modificaciones de última hora demasiado complejas.

  El objetivo de este seguimiento es, nuevamente, mejorar futuras planificaciones, por lo que este sistema de seguimiento de tareas debería disponer de su correspondiente sistema de análisis de tareas.

  Esta característica formaba parte de las especificaciones del sistema *Brújula.*

4 Conclusiones

Este proyecto se ha desarrollado en su mayor parte durante un periodo de prácticas en empresa, lo que me ha servido como un interesante primer contacto con el mundo laboral, permitiéndome conocer de primera mano un futuro entorno de trabajo. Además, el hecho de que haya estado enfocado hacia la gestión de proyectos me ha dado una visión muy detallada de cómo se realiza, de cuál es su problemática habitual y de las técnicas de mejora que se pueden aplicar.

Por otro lado, el haber dispuesto de un sistema de código abierto ha sido una gran ayuda para poder realizar prototipos de una forma rápida y sencilla, permitiendo validar los requisitos con el *cliente* en muy poco tiempo. Lógicamente al tratarse de un cliente con larga experiencia en el desarrollo de software la recogida de requisitos ha sido más sencilla de lo habitual, aunque en mi opinión ha sido adecuado como primer contacto.

Este primer contacto con el desarrollo de software en un *entorno real* ha servido para poner en práctica los conocimientos obtenidos durante mi formación, en especial en lo que al campo de la Ingeniería del Software se refiere. De hecho, la realización de este proyecto me ha permitido seguir todo el ciclo de vida desde la recogida de las necesidades del cliente hasta la implantación de un software que las cubriera.

Como conclusión final, este proyecto me ha servido para conocer tanto las ventajas como las desventajas de los proyectos software libre; Las ventajas de disponer de un código de distribución y modificación libre son de sobra conocidas, pero para obtener un producto de calidad es vital que exista una fuerte motivación moral, económica o académica en los desarrolladores.

Bibliografía

- [dP 2008] Página web oficial de dotProject, desde la que se puede descargar el programa, otros módulos desarrollados para él, traducciones y acceder a la documentación existente a nivel de usuario. La documentación técnica es accesible principalmente a través de foros y del *Priority Support*.
  http://www.dotproject.net/
  Enlace consultado por última vez en febrero de 2008.

- [PHProjekt 2008] PHProjekt es la principal alternativa de código abierto a dotProject. Desde este enlace es posible descargar la plataforma, nuevos módulos, traducciones y otros.
  http://www.phprojekt.com/index.php?&newlang=eng
  Enlace consultado por última vez en febrero de 2008.

- [IR 2005] Apuntes de la asignatura Ingeniería de Requisitos, curso 2005/2006, Facultad de Informática de la Universidad de Murcia, del profesor Ambrosio Toval. Han sido de utilidad como elemento de consulta general.

- [Glass 2002] Libro recomendado en el recurso anterior para describir la importancia de una buena recogida y seguimiento de los requisitos del sistema.
  Glass, R. L. (2002). Software Engineering: Facts and Fallacies, Addison-Wesley.

- [Blackburn 1996] Libro recomendado por el mismo recurso para describir la importancia de una buena recogida de requisitos a partir de la experiencia de ciertas empresas.
  Blackburn, J.D. et al. (1996). Time-Based Software Development.

- [Letelier 2008] Recurso que recoge una aproximación a la trazabilidad de requisitos. Considero que es una opción excesivamente compleja como para tener la adecuada aceptación por parte del usuario final. Por Patricio Letelier, Departamento Sistemas Informáticos y Computación de la Universidad Politécnica de Valencia.
  https://pid.dsic.upv.es/C1/Material/Documentos%20Disponibles/Trazabilidad.ppt
  Enlace consultado por última vez en febrero de 2008.

- [PSI 2008] Este recurso agrupa un conjunto de conceptos específicos sobre los requisitos y su trazabilidad. Son interesantes, especialmente como fuente de ideas ante un sistema de trazabilidad más complejo. Asignatura de Ingeniería del Software 1, Universidad Carlos III de Madrid.
  http://www.ie.inf.uc3m.es/grupo/docencia/reglada/psi/unidad7-DOC.pdf
  Enlace consultado por última vez en febrero de 2008.

- [SWEBOK 2004] Guía del SWEBOK, empleada como recurso de consulta esporádico, edición de 2004.
  http://www.swebok.org/
  Enlace consultado por última vez en febrero de 2008.

Anexos

I. Integración de dotProject y RequisitePro

II. Metodología de trabajo en Versas

III. Plantilla de Acta de Reunión

IV. Implantación de dotProject en SQA

V. Otros Módulos para dotProject

VI. Tutorial “proyectos SQA para dotProject”

VII. Manual de programación en dotProject

I. Integración de dotProject y RequisitePro

Los mecanismos de extensibilidad a nivel de interfaz de usuario de dotProject consisten en la programación de módulos empleando cualesquier lenguajes soportados por el servidor web en el que esté instalado. Por otro lado, la base de datos que utiliza internamente es MySQL, de modo que cualquier programa con una interfaz compatible con ella podrá importar y exportar datos a este entorno.

Por su parte, los mecanismos que facilita RequisitePro son muy amplios y flexibles, como se resume a continuación:

- **Integración con la interfaz de usuario**

RequisitePro permite incluir nuevos mecanismos en la interfaz gráfica de usuario del entorno, por lo que sería posible la importación desde una base de datos externa, siempre que existiera compatibilidad con las bases de datos soportadas.

- **Programación de componentes específicos**

Existe una API llamada *reqpro.dll* cuyos procedimientos están recogidos en la ayuda de RequisitePro. Para ello se emplea la *Interfaz de Extensibilidad* (*Extensibility Interface*) en lenguaje VisualBasic, aunque es posible emplear C o Java. En el siguiente enlace se muestra cómo crear unos primeros scripts que personalicen la herramienta de gestión de requisitos:

http://www.ibm.com/developerworks/rational/library/445.html

El siguiente enlace muestra cómo realizar scripts para la administración de proyectos:

http://www.ibm.com/developerworks/rational/library/07/1225_kuriyala/

En el artículo anterior se explica cómo modificar:

- Los permisos de acceso a proyectos
- Las propiedades de un proyecto
- Las propiedades y atributos de cada requisito
- La trazabilidad entre componentes de un proyecto

En el siguiente foro se trata este apartado en profundidad:

http://www-128.ibm.com/developerworks/forums/forum.jspa?forumID=323&start=0

- **Integración con la base de datos**

RequisitePro permite acceder a la base de datos de requisitos a través de las siguientes interfaces:

  - Microsoft Access
  - SQL server
  - Oracle
  - DB2

DotProject es compatible con MySQL, no existiendo constancia entre la documentación pública de que internamente se admita el uso de otros sistemas de bases de datos. Por otro lado, como se comenta en uno de los hilos del foro *developerWorks*, RequisitePro no se diseñó para ser compatible con MySQL; La compatibilidad con este sistema no está recogida ni hay intención de recogerla.

Conclusión

Aunque RequisitePro provee de todo tipo de mecanismos de extensibilidad, no sería posible que dotProject y RequisitePro compartieran una única base de datos interna. Las alternativas que surgen en este punto se limitan a emplear mecanismos de importación/exportación de cada una de las bases de datos a través de las interfaces de ambos entornos.

II. Metodología de trabajo en Versas

Introducción

En este anexo se realiza una breve descripción de la metodología de trabajo de uno de los departamentos de la empresa SQA que prácticamente puede ser extendido a los otros dos. Ha sido elaborado a partir de distintas observaciones y entrevistas durante mi periodo de prácticas.

Esta descripción ha sido descompuesta tomando como referencia las etapas de un ciclo de vida de software con las siguientes etapas:

1. Presentación del proyecto al cliente
2. Presupuestación del proyecto
3. Entrevista con el cliente
4. Catalogación de requisitos
5. Análisis
6. Diseño
7. Programación
8. Implantación y mantenimiento

A continuación se describe cada una de ellas.

1. **Presentación del proyecto**

Consiste en la presentación de las posibilidades de la plataforma Versas al cliente. Se realiza en persona y en algunos casos ayudados por presentaciones en formato *Microsoft Powerpoint*. Concretamente se establece:

- El contrato de software
- Las características del proyecto
- Las condiciones de desarrollo y modificaciones necesarias

2. **Presupuestación del proyecto**

Consiste en acordar con el cliente la cantidad a pagar y las condiciones del proyecto:

- En los proyectos que no requieran modificaciones a la plataforma los costes están dependen del número de licencias que solicite el cliente.

- En proyectos para los que el cliente solicite alguna nueva funcionalidad o modificación, también se tienen en cuenta los costes de desarrollo asociados. Para ello es necesaria una estimación de costes a priori.

- Un tercer caso son los proyectos que requieren la adaptación a otros sistemas existentes, en los que también será necesario estimar los costes a priori previo análisis del sistema anterior.

El presupuesto puede estar sujeto a políticas de descuento diferentes según convenio.

La política de la empresa es ofrecer un producto *llave en mano*, por lo que dentro del presupuesto se incluyen todos los servicios necesarios para la correcta implantación del sistema.

Además de cuestiones económicas, esta etapa incluye la negociación de las condiciones y plazos de entrega, formación, adquisición de hardware, soporte técnico y mantenimiento del sistema implantado. Como en ocasiones las funcionalidades solicitadas por el cliente no son las esperadas, ambas partes se pueden ver en la necesidad de renegociar el presupuesto.

No hay un criterio objetivo para establecer el coste de desarrollo (en tiempo) de una determinada funcionalidad, modificación o adaptación, ya que no se utiliza ningún tipo de métrica formal; El analista es el encargado de realizar esta estimación a partir de su experiencia y conocimiento de la plataforma.

En esta etapa sólo se recoge de forma informatizada el presupuesto (en formato *Microsoft Word* o *Adobe PDF*) que será firmado posteriormente de forma manuscrita. En alguna ocasión se estiman los costes en hojas de cálculo a modo de borrador.

3. Entrevistas con el cliente

En las condiciones del contrato de software se acuerda un número de entrevistas a las que ambas partes se comprometen a asistir (entrevistas previstas), aunque dependiendo de cómo se lleve a cabo el desarrollo también se podrán realizar otras no previstas. En ambos casos su objetivo es:

- Conocer la metodología de trabajo en el entorno de aplicación.
- Describir las funcionalidades o cambios que necesita el ciente.
- Comprobar la aceptación de las funcionalidades ya implementadas.

Suelen ser verbales, con lo que no quedan registradas en ningún soporte digital (salvo alguna excepción en formato audio). Tras ellas se redacta, con más o menos detalle, un "acta de reunión" en la que queda recogido todo lo discutido en la entrevista. La mayoría de la documentación obtenida en esta etapa se recoge únicamente en papel.

Las actas están divididas en las siguientes secciones:

- Objetivos por los que se convoca la reunión (orden del día), descrito en una línea.
- Desarrollo, en el que se describe detalladamente tras la reunión la información concluida tras la misma.
- Requisitos identificados. Se extraen los requisitos que deben ser implementados y se numeran.
- Acuerdos a los que han llegado las partes tras la reunión. Suele tratarse de cuestiones sobre el cumplimiento contractual de las partes.
- Tareas pendientes y por realizar por parte de empresa y cliente (sin ningún compromiso de tiempo). No guarda relación con las "tareas" que definirán la implementación.
- Listado de los documentos entregados por el cliente, tanto en formato digital como en papel.
- Firma de conformidad entre cliente y empresa. Aunque el documento debe estar firmado por cliente y analista para tener validez, suele existir cierta desidia por parte del cliente por lo que estas actas no siempre se firman.

Los requisitos que se describen en estos actas no siempre se corresponden con el concepto habitual de requisito, sino que puede consistir en describir la tarea que lo implementaría. De hecho, muchos de los requisitos se redactan utilizando expresiones como “hay que modificar...”, “hay que hacer...” o similares, buscando con esto explicar el trabajo a realizar en un lenguaje cercano al cliente. En la práctica, los requisitos no se formalizan hasta la redacción de la tarea.

Una variante de estas actas de reunión son la partes de incidencias. Básicamente su función es describir los errores encontrados durante la ejecución de una aplicación ya entregada o durante su implantación. Rara vez se redactan.

4. Catalogación de requisitos

Los requisitos no siempre se recogen dentro de las actas. En ocasiones aparecen en hojas de cálculo independientes, normalmente agrupados por los módulos funcionales a los que deberían pertenecer. Además, especialmente en los proyectos en los se producen modificaciones constantes, no se llegan a recoger de ningún modo, de forma que las modificaciones pasan directamente a la sección de desarrollo. Normalmente se registran en las situaciones en las que se debe realizar una nueva presupuestación para el cliente.

Una catalogación de todos los requisitos de la plataforma Versas sería inviable debido a su complejidad.

5. Análisis

Se considera dentro de la recogida de requisitos.

6. Diseño

Básicamente consiste en redactar tareas para la fase de desarrollo a partir de los requisitos identificados. Para ello se realizan esquemas en papel y reuniones entre analistas donde se discute cómo abordar cada requisito, por lo que no es frecuente encontrar documentos informatizados sobre esta etapa.

El producto de esta etapa es un listado de tareas asignadas a distintos desarrolladores junto a una estimación subjetiva del tiempo requerido para cada una.

7. Programación

Consiste en realizar las tareas encomendadas al personal de desarrollo. Desde hace poco tiempo se está llevando a cabo una asignación de tareas y seguimiento del trabajo de cada programador a través de *Microsoft Access*, debido a cuestiones de presupuestación por revisión o inclusión de nuevos requisitos con total implicación del personal en este aspecto.

8. Implantación/mantenimiento

Consiste en la recogida de las incidencias acaecidas durante estas etapas a través de partes de incidencia o actas de reunión. Teóricamente se debería revisar todo el proceso de desarrollo desde la recogida de requisitos pero, como ya se ha comentado, estas modificaciones rara vez se registran, pasando directamente de la entrevista con el cliente a la fase de programación.

La forma de recibir las incidencias detectadas por el cliente varía desde la comunicación verbal hasta el envío de capturas de pantalla mostrando la situación de error junto a su explicación por escrito.

Las posibles incidencias que se suelen registrar son:

- Errores de programación encontrados durante la explotación o implantación del sistema.
- Fallos o pérdida de hardware.
- Funcionalidades no implementadas correctamente.
- Cambios o nuevas funcionalidades solicitadas por el cliente.
- Seguimiento del proceso de implantación.

III. Plantilla de Acta de Reunión

Ver documento ANEXO - *Plantilla de Acta de Reunión.pdf*

IV. Implantación de dotProject en SQA

En el momento de presentar este Proyecto Fin de Carrera la empresa llevaba dos meses utilizando dotProject para la planificación de sus proyectos. El nombre interno que la empresa ha elegido para este sistema es SQAdra.

El estado de la implantación por departamentos en febrero de 2008 se resume a continuación:

- El departamento *Versas* está trabajando en adaptar a dotProject el registro de tareas que estaban llevando en *Microsoft Access* desde hace unos meses. Esta adaptación probablemente implicará alguna pequeña modificación del módulo original *Tasks*, bien a través de la edición directa de este módulo o bien a través de una copia del mismo. Mientras tanto se está empleando dotProject para el registro de las nuevas tareas.

- En el departamento *Neweb* se ha decidido iniciar la implantación del sistema de gestión de proyectos en cuanto reciba las últimas modificaciones. Este departamento está muy interesado en el módulo de requisitos, lo que ayudará a obtener una certificación de calidad *ISO*.

- En el departamento *Gobierno* se ha implantado con éxito, siendo el departamento que más proyectos tiene registrados hasta la fecha.

Por otro lado, también existe un interés en realizar mejoras generales a la plataforma. Estas modificaciones se realizarán sobre los módulos originales de la plataforma o sobre copias de algunos de los existentes en función de la complejidad de la modificación, como por ejemplo mejorar los mecanismos de navegabilidad.

Dentro de estas mejoras se incluye un filtrado de los listados por departamentos. Hasta ahora se ha conseguido *simular* este filtrado empleando el modelo de permisos, bloqueando el acceso de cada usuario a todos los proyectos que no son de su departamento. Este proceso sería muy costoso de realizar manualmente por lo que se ha optado por implementarlos a través de *triggers*. Aunque este sistema no implica modificaciones sobre el código original, requiere demasiadas comprobaciones en cada acceso a un módulo, lo que sobrecarga excesivamente el sistema. Esta técnica será sustituida por otra basada en modificaciones sobre el código original o sobre copias de módulos existentes.

El seguimiento de proyectos se complementa con la generación de informes basados en los que dotProject incluye inicialmente. Con estos informes se pretende identificar la causa de cada desviación en los costes de un proyecto durante su desarrollo a través de un análisis adecuado. Posteriormente se elaborarían mecanismos para evitar que las desviaciones se vuelvan a producir.

Por último, se están empleando plantillas para la creación de nuevos proyectos, tanto para los nuevos desarrollos como para los desarrollos en curso, encontrándose estas plantillas en proceso de refinamiento.

V. Otros módulos para dotProject

Introducción

En este anexo se describen todos los módulos disponibles a 6 de noviembre de 2007 en la sección de *addon*s de dotProject en SourceForge.net. Sorprende la baja calidad de los mismos (en su mayoría están inacabados), especialmente en cuanto a compatibilidad y facilidad de instalación se refiere. De hecho, la documentación existente para cada uno es muy escasa o nula en casi todos los casos, desanimando a terceros a continuar el trabajo hacia una versión estable.

A continuación se detalla la funcionalidad de todos los módulos publicados:

Backup Module v2.0 (2005.04.08)
Proviene de una versión adaptada de dotProject 1.0 a 2.0. Se emplea para realizar copias de seguridad de la base de datos. Tras su instalación en nuestra versión de dotProjet (2.1) había problemas para restaurar los datos desde la copia de seguridad, se emplease el formato que se emplease (.sql, .xml o .zip).

Import Export v0.2 (2007.10.10)
Nueva versión del módulo *Backup* que permite exportar el contenido de la base de datos a los formatos *.xml, .sql, .csv, .vcf* y *MSProject* con opción de compresión. En la versión de dotProject que empleamos da error al:

- no especificar ningún fichero a importar.
- exportar en cualquier formato que no sea *.sql*
- importar en cualquier formato.

No existe ninguna documentación sobre este módulo (ni siquiera un *readme*), por lo que se descarta un análisis en mayor profundidad de este módulo.

Eventum Integration Module v1.1.5 (2005.09.11)
Integración entre dotProject y el módulo de Eventum de MySQL. Es necesario disponer de dicho módulo para utilizarlo. No ha sido probado.

Finance v0.1 (2007.10.09)
Control de ingresos y gastos de un proyecto determinado. Permite registrar facturas e ingresos, pero curiosamente no permite eliminarlas si se registran por error. No es compatible con el módulo *Invoice* (explicado más adelante).

Helpdesk (2004.05.07)
Versión anterior del módulo *Tickets* para la versión 1.0, posteriormente incluida en las versiones 2.0 y 2.1.

Inventory Module v0.2 (2003.12.21)
Se emplea para registrar las adquisiciones de materiales de una empresa. Además de tratarse de una versión *alfa* fue desarrollado antes de la salida de dotProject 2.0, por lo que es incompatible con nuestra instalación.

Invoices v0.4 (2006.06.05)
Permite emplear *log*s de tareas como parte de una factura, llevando un seguimiento de la misma hasta que es abonada. No funciona sobre dotProject v2.1.

Journal Module v1.0 (2004.03.29)
Es una modificación del módulo *History* incluido en la distribución inicial de dotProject como módulo *de usuario* en lugar de *core*. Este módulo permitía añadir anotaciones a cada proyecto a través de pestañas en otros módulos. El módulo *Journal* parece ser un primer intento de módulo para dotProject de algún programador, por lo que no aporta nada nuevo.

Project Designer v1 (2007.3.15)
Es una revisión de la *vista de proyecto* original de dotProject, configurada como un módulo independiente. Permite crear tareas para el proyecto de forma rápida y sencilla, mostrando la gráfica de Gantt del proyecto actualizada tras su creación. También facilita la edición de sus atributos, adjuntar ficheros y crear distintos informes listos para imprimir con un solo click. Es el el módulo tipo *addon* con mayor utilidad práctica. En la documentación se dice que pasará a formar parte de la versión 3 de dotProject.

Risks Module v2.2 (2005.07.08)

Módulo para asignar y controlar riesgos de proyectos y tareas. Para cada riesgo se define:

- Nombre, descripción y comentarios.
- Probabilidad de que suceda
- Impacto en horas
- Propietario

Adolece de algunos fallos, especialmente durante la creación/edición de riesgos y además se echa en falta algún método para agregar riesgos a proyectos y tareas desde sus respectivas vistas. Tampoco incluye ningún modelo de informe o similar para analizar los riesgos registrados. Sin embargo parece una buena base para un refinamiento posterior.

Ticketsmith (2006.06.04)

Viene incluido entre los módulos *core* de la versión 2.1 de dotProject y por algún motivo se ha incluido como *addon*. Está desarrollado por una *thirdparty* que impone condiciones de uso muy restrictivas por lo que no va a pasar a formar parte de dotProject v3. Como ya se indicó en el documento principal, se trata de un sistema de resolución de incidencias a través de *tickets*.

Unicost v1.0 (2006.06.21)

Su instalación requiere modificar los módulos originales de dotProject manualmente (aunque no se dice cómo), por lo que hemos desestimado su uso.

VI. Tutorial “proyectos SQA para dotProject”

Introducción

En este trabajo se muestran los pasos seguidos para desarrollar el módulo Proyectos SQA, que en su momento consistió en una muestra de las capacidades de la plataforma dotProject. Se realizó partiendo del código del tutorial disponible en la página oficial de dotProject, que es la metodología de desarrollo recomendada por sus autores.

En el documento llamado “Manual de programación en dotProject” se ha descrito cómo y dónde encontrar documentación relativa a dotProject, si bien este documento muestra técnicas que no se muestran en él.

Primeros pasos

Antes de empezar el tutorial será bueno echarle una ojeada al tutorial que han puesto los desarrolladores de dotProject para explicar cómo programar módulos a modo de addons. El módulo de ejemplo se llama “Einstein” y se puede encontrar aquí:

http://sites.sakienvirotech.com/dp_docs/tiki/tiki-index.php?page=Writing+a+Module

Al parecer estaba pensado para funcionar en la versión 1.0 de dotProject. Como utilizamos dotProject 2.1 aparecen dos errores a la hora de probar el código de ejemplo, que según el foro oficial se corrigen así:

- Cambiar lo siguiente en el index.php, línea 42:

  $tabBox = new CTabBox("?m=einstein", "{$dPconfig['root_dir']}/modules/einstein/", $tab);

- Y en el addedit.php, reemplazar la línea 108 por:

  <textarea name="einstein_quote" cols="60" style="height:100px; font-size:8pt">
  <?php echo dPformSafe($obj->einstein_quote);?> </textarea>

A modo de resumen, vemos que los pasos a seguir para crear un módulo son:

1. Crear un directorio con el nombre del módulo. Las imágenes e iconos se deben guardar en un directorio **/images** dentro de ese mismo directorio.

2. Desarrollar un script de instalación (**setup.php**) para que el administrador de módulos lo reconozca, utilizando el setup.php del tutorial como plantilla. A partir de este momento podremos instalarlo como cualquier otro módulo y verlo dentro del entorno dotProject en su propia pestaña.

Una vez instalado y probado este módulo, pasamos a explicar cómo adaptarlo a nuestras necesidades.

Creación del módulo

A la hora de abordar una modificación del programa original podíamos pensar en modificar alguno de los módulos existentes sin más. Sin embargo nos gustaría que nuestro trabajo fuera utilizable por cualquier otro usuario de dotProject y compatible con futuras versiones de la plataforma. Por tanto, dado que dotProject nos facilita mecanismos para crear módulos externos, que incluso podrán integrarse dentro de los existentes, se ha realizado un módulo independiente. Como ya se ha comentado, tomaremos como base el “módulo einstein” que se describe en el tutorial “oficial” de dotProject.

Empezaremos por cambiar el nombre del módulo, versión, descripción, etc. Para ello basta con editar el fichero setup.php así:

```
// MODULE CONFIGURATION DEFINITION
$config = array();
$config['mod_name'] = 'Proyectos SQA';       // name the module
$config['mod_version'] = '0.0.0';            // add a version number
$config['mod_directory'] = 'sqa_proyectos';  // tell dotProject where to find this module
$config['mod_setup_class'] = 'CSetupSqaProyectos';      // the name of the PHP setup class
$config['mod_type'] = 'user';                // 'core' for modules distributed
                                             // with dP by standard, 'user' for
                                             // additional modules from dotmods
$config['mod_ui_name'] = 'SQA proyectos';    // the name that is shown in the
                                             // main menu of the User Interface
$config['mod_ui_icon'] = 'communicate.gif';  // name of a related icon
$config['mod_description'] ='SQA - Módulo propio de seguimiento de costes de proyectos';
$config['mod_config'] = false;               // show 'configure' link in viewmods
```

IMPORTANTE: El atributo mod_type **siempre** debe valer “user” (lo digo por experiencia). No recomiendo experimentar con esto.

El significado de cada campo es bastante intuitivo. Recomiendo por claridad utilizar el mismo criterio a la hora de nombrar y renombrar variables que el que se usa habitualmente en otros módulos, como por ejemplo CSetupNombreMódulo. Esto además ayuda a encontrar clases, atributos y demás “por intuición”, es utilizado en algunas situaciones por dotProject para interpretar el contenido de los elementos del módulo. También es aplicable a la hora de nombrar otros “elementos sensibles” de nuestro código (clases, variables, atributos de campos en SQL, etc).

Lo siguiente será renombrar variables dentro de todos los demás ficheros del módulo del tutorial. En concreto, para cambiar el directorio del proyecto debemos modificar:

- El parámetro mod_directory (como se muestra arriba)

- Cada link que aparezca en el código de cualquier fichero, cuyos argumentos serán de la forma:
  m=modulo_proyecto

- En cada ruta de directorio que aparezca en el código de cualquier fichero, cuyos argumentos serán de la forma:
  /modules/modulo_proyecto

Ya que la técnica más habitual será la “reutilización” de código (no olvidemos que se trata de un proyecto de código abierto) el renombrado de variables será bastante frecuente, por lo que es aconsejable que nuestro editor nos de facilidades para ello.

Cambiar el nombre del proyecto implica cambiar el nombre de la clase que usaremos para realizar nuestras consultas a la base de datos. DotProject carga al inicio una clase de nombre nombreMódulo.class.php, a través de la que podremos asignar atributos a cada tabla.

En este momento podremos instalar nuestro nuevo módulo a través del administrador de módulos, que aún tendrá la funcionalidad del módulo del ejemplo pero al menos será reconocido como un módulo independiente.

Nombrado de ficheros

Es obligatorio que exista un index.php, un setup.php y un módulo.class.php para que el entorno reconozca el módulo. De igual modo que sucede en otros módulos, se sigue un cierto convenio a la hora de nombrar los demás ficheros (y que recomiendo fervientemente seguir):

- un **addobjeto.php** o **addedit.php**, para añadir o editar cada objeto.
- un **view.php** para ver ese objeto, opcionalmente.
- varios **vw_pestaña.php**, uno por cada pestaña que se verá al abrir nuestro módulo en la GUI de dotProject. Normalmente contendrán listados de objetos con más o menos información adicional.
- un **do_tabla_aed.php** para definir la interfaz entre la clase CTabla y la base de datos (una o varias por módulo). Las clases también siguen un convenio de nombrado que hay que respetar para que dotProject lo reconozca.
- un **index.html** que será el que defina cuándo y cómo ver cada pestaña. Se recomienda “reutilizar” el código de algún módulo existente.
- un directorio **/images**, si el módulo tiene imágenes o iconos propios.
- hay otros “nombres especiales” para poder añadir pestañas en los módulos que trae por defecto dotProject (como tasks o projects). Se explica en el documento “manual de programación en dotProject”.

Manipulando la base de datos

En el módulo Proyectos SQA se hay dos tablas que pueden servir como ejemplo:

- Costes, que nos permitirá llevar un seguimiento detallado de cada gasto de la empresa, indicando a qué tarea y/o a qué proyecto corresponde. Se incluye el importe y una descripción textual.
- Tarifas, que son un primer paso para llevar un registro del tiempo trabajado y del tipo de trabajo realizado. Definen un intervalo de tiempo durante el que cada tarifa es válidas.

Como se comentó en la sección anterior, debemos indicar en el nombreMódulo.class.php los atributos de cada tabla. Por ejemplo, para la tabla costes (llamada por convenio CCostes) se muestra la declaración sus atributos, de un constructor y de un destructor:

```
class CCostes extends CDpObject {
        // link variables to the object (according to the existing columns in the
        // database table)
        var $coste_id = NULL;           //use NULL for a NEW object, so the database
                                        // automatically assigns an unique id by 'NOT
                                        // NULL'-functionality
        var $coste_concepto = NULL;
        var $coste_valor = NULL;
        var $coste_proyecto = NULL;
        var $coste_tarea = NULL;

        // the constructor of the CCostes class, always combined with the table name
        // and the unique key of the table
        function CCostes() {
                $this->CDpObject( 'costes', 'coste_id' );
        }
        // overload the delete method of the parent class for adaptation for table's
        // needs
        function delete() {
                $sql = "DELETE FROM costes WHERE coste_id = $this->coste_id";
                if (!db_exec( $sql )) {
                        return db_error();
                } else {
                        return NULL;
                }
        }
}
```

Con respecto al módulo Einstein basta con renombrar variables y añadir los campos que queramos. Vemos que no es necesario asignar un tipo a los atributos, porque el tipo se hereda de la base de datos. Las funciones CTabla() y delete() son el constructor y destructor de cada objeto que insertemos/eliminemos de la base de datos. En el constructor no será necesario cambiar nada más que el renombrado, pero en el destructor puede que necesitemos añadir algún delete extra si nuestras tablas tienen campos multivaluados, por ejemplo.

Como se puede ver, existe otro convenio de nombres a la hora de dar valores a los campos de elementos en MySQL. Las tablas normalmente se denominarán en plural y cada atributo de cada elemento de la misma con el nombre de la tabla en singular seguido del nombre del atributo. Esto será extremadamente útil para deducir los atributos de otras tablas sin necesidad de recurrir a la documentación de dotProject. Por ejemplo project_name, task_name, company_name, etc, o vínculos entre tablas como task_project, project_company, etc.

El siguiente paso es realizar las operaciones de creación de tablas en la base de datos. Para ello editaremos el fichero setup.php que se ejecutará al instalar o desinstalar el módulo. Por tanto, si realizamos modificaciones en la base de datos durante el proceso de desarrollo del módulo, debemos desinstalarlo, modificar el código en todos los ficheros implicados, y reinstalarlo para que los cambios tengan efecto (aunque también es posible editar la base de datos por separado...). Como la creación y destrucción de tablas se lleva a cabo a través de consultas en MySQL, es relativamente sencillo crear tablas muy complejas o con elementos cruzados entre ellas.

Nuestro módulo de ejemplo contiene una clase que por “convenio dotProject“ se llama CsetupMódulo, cuyo código contiene (ver página siguiente):

- Una función de instalación en la que se crean las tablas usando código SQL. Respecto a la tabla costes:

```
function install() {
        $sql = "CREATE TABLE costes ( " .
                "  coste_id int(11) unsigned NOT NULL auto_increment" .
                ", coste_concepto text" .
                ", coste_valor decimal(10,2) default '0.00'".
                ", coste_proyecto int(11)".
                ", coste_tarea int(10)" .
                ", PRIMARY KEY(coste_id)" .
                ", UNIQUE  KEY(coste_id)" .
                ", FOREIGN KEY(coste_proyecto) REFERENCES
                projects(project_id)".
                ", FOREIGN KEY(coste_tarea) REFERENCES tasks(task_id)".
                ") TYPE=MyISAM;";
        db_exec( $sql ); db_error();          // execute the queryString
        return null;
}
```

- Una función de desinstalación, en la que se eliminan las tablas SQL. En nuestro caso basta con escribir:

```
function remove() {                              // run this method on uninstall process
        db_exec( "DROP TABLE costes;" );// remove the table from database
        db_exec( "DROP TABLE tarifas;" );// remove the table from database
        return null;
}
```

- También existe una función de actualización de versiones, pero no la hemos implementado.

Manipulación de elementos de la base de datos

Ahora vamos a explicar cómo leer y modificar los campos de la base de datos. Por un lado tenemos que crear un do_tabla_aed.php, que será la interfaz entre la clase CTabla y la base de datos. En principio basta con renombrar variables de cualquier fichero original de dotProject, aunque si utilizásemos algún atributo complejo (como multivaluados) probablemente necesitemos añadir alguna que otra operación con MySQL.

A continuación se muestran algunas operaciones básicas con MySQL:

- Visualización de tablas:

  En el caso de la visualización de tablas el proceso más común es el de realizar una consulta SQL, almacenar el texto en una variable, y ejecutarla con alguna de las funciones que nos facilita dotProject. En el caso de la tabla costes hay referencias a las tablas originales projects y tasks, por lo que la consulta se deberá realizar como sigue:

```
$sql = "    SELECT coste_concepto, coste_valor, project_name, task_name
            FROM costes, projects, tasks
            WHERE project_id=coste_proyecto and task_id=coste_tarea";
$costes = db_loadList( $sql );
```

  Hay mucha documentación sobre SQL por Internet, así que no veo necesario explicar esto en más profundidad. Recomiendo utilizar por ejemplo phpMyAdmin (incluido en XAMPP) para practicar antes de incluir cualquier consulta en el código. También recomiendo ver el contenido de la variable que resulta de la consulta utilizando la función de php print_r($variable).

  El procedimiento habitual para mostrar el resultado de la consulta en forma de lista será el que se muestra a continuación:

```
foreach ($costes as $row) {
            ...
            <td ><?php echo $row["coste_concepto"];?></td>
            <td ><?php echo $row["coste_valor"];?></td>
            <td ><?php echo $row["project_name"];?></td>
            <td ><?php echo $row["task_name"];?></td>
            ...
}
```

  El hecho de que se utilicen consultas SQL nos permitirá que el listado pueda contener valores de otras tablas, que esté ordenado, o que contenga valores calculados.

- Edición de tablas:

  En el caso de la edición de tablas es importante tener claro de antemano cómo va a ser la interfaz, qué campos contendrá, su tipo, los vínculos entre ellos y las restricciones a la hora de introducirlos.

  Para definir la posición de cada elemento en la pantalla recomiendo diseñar la tabla en la que irá el formulario utilizando algún editor html. Después copiaremos la estructura en texto a nuestro código php y añadiremos campos libremente.

  Como ejemplo, en addcoste.php y addtarea.php hemos utilizado:

  - Campos para introducir texto.
  - Campos para introducir cantidades monetarias.
  - Campos para elegir filas de otras tablas.
  - Campos para introducir fechas.

  Para recuperar la información de cada campo del objeto debemos crear un objeto de la clase CTabla utilizando la función load(). Esto nos creará en la variable que indiquemos una estructura de datos con todos los atributos del objeto, de modo que podamos recuperar su valor en cada campo de nuestro formulario de edición como se explicará después.

  En principio, utilizaremos el mismo formulario para la creación y la edición de objetos. En el primer caso llamaremos al formulario con un identificador mayor que cero, que se corresponderá con la clave primaria del objeto que queremos editar. En el segundo caso lo llamaremos con un identificador igual a cero, ya que no se permitirá este valor como clave primaria de ningún elemento de la base de datos.

  Aquí tenemos un ejemplo de load:

```
$obj = new CCostes();
if (!$obj->load( $coste_id, false ) && $coste_id > 0) {
// show some error messages using the dPFramework if loadOperation failed
// these error messages are nicely integrated with the frontend of dP
// use detailed error messages as often as possible
        $AppUI->setMsg( 'Costes' );
        $AppUI->setMsg( "invalidID", UI_MSG_ERROR, true );
        $AppUI->redirect();                    // go back to the calling location
}
```

Inicializaremos los campos con el valor que tuviera el atributo en el momento de cargarlo utilizando:

```
echo $obj->coste_concepto;
```

Después el usuario modificará el valor, pulsará “enviar” y se enviará el formulario con el valor modificado. DotProject, a través de do_tabla_aed.php, se encargará de que el objeto modificado se actualice en la base de datos con los nuevos valores.

Para visualizar el contenido de los atributos utilizaremos el tipo de campo que más se ajuste al dato que queremos introducir:

- En el caso de de **texto plano** basta con darle el nombre que el atributo tendrá en la base de datos e indicar el campo que recibirá el valor:

```
<textarea name="coste_concepto" cols="40" style="height:14px;
        font-size:8pt"><?php
        echo dPformSafe( $obj->coste_concepto );
?></textarea>
```

  El nombre del área de texto deberá ser el mismo que el del atributo del objeto, pues los dos hacen referencia al mismo campo en la base de datos (en este caso coste_concepto).

  Al ver este tutorial después de haber trabajado unos meses con dotProject, me he dado cuenta que el código utilizado en el módulo Einstein se salta a la torera todos los convenios que aparecen en la (por otro lado escasa) documentación de dotProject, probablemente por haber sido desarrollado para alguna de las primeras versiones de la plataforma.

  En concreto existen plantillas xml para definir el tamaño, fuente y demás atributos de los textos que aparecen en pantalla, y es tan simple como utilizar, en lugar del código que aparece ahí arriba:

```
<input type="text" name="coste_concepto" size="22" maxlength="20" class="text" value="<?php
        echo dPformSafe( $obj->coste_concepto );
?>"/>
```

  El atributo size define el ancho del campo de datos, mientras que maxlength limita el número de caracteres que podemos introducir en él.

Para el caso de textos de varias líneas (con barra de scroll a la derecha) usaríamos algo así:

```
<textarea name="coste_descripcion" cols="50" rows="6" class="text"><?php
        echo dPformSafe( $obj->coste_descripcion );
?></textarea>
```

Esto nos garantiza que se vea correctamente en Explorer y Firefox.

- En el caso de introducir **cantidades monetarias** no se ha realizado ningún control de errores en el módulo de ejemplo, por lo que cualquier cadena de caracteres que introduzcamos será interpretadas directamente por la base de datos. Se podría haber controlado que este valor fuera correcto una vez el usuario pulse “enviar” a través de un script en javascript. En el documento “Manual de programación en DotProject” se explica cómo hacerlo.

- Para introducir valores a través de **menús desplegables**, en primer lugar hay que recuperar los valores de la tabla que los contenga (aunque cualquier array asociativo de PHP nos valdría). Por ejemplo, para acceder a la tabla proyectos utilizamos este código:

```
//accederemos a projects para enlazar el proyecto al coste
require_once( $AppUI->getModuleClass ('projects' ) );
$aux = new CProject();
$projects = $aux->getAllowedRecords( $AppUI->user_id, 'project_id,project_name','project_name' );
$projects = arrayMerge( array( '0'=>'' ), $projects );
```

Vemos que hemos utilizado la función getAllowedRecords del módulo projects. Sin embargo esto no siempre será posible porque habrá módulos (la mayoría) que no tengan implementada esta función y tendremos que realizar una consulta SQL de toda la vida. Es más, una consulta SQL nos hubiera dado más posibilidades, como ordenar los campos por identificador o alfabéticamente. Sin embargo, usando getAllowedRecords dotProject realiza un control de permisos de acceso, que en nuestro caso nos resulta indiferente.

La última línea (arrayMerge) se utiliza para introducir un espacio en blanco en la primera posición del desplegable, de modo que podamos dejar el valor en blanco.

Por último, para utilizar el desplegable en el formulario, se utiliza la función “arraySelect”:

```
<td><?php
    echo arraySelect( $projects, 'coste_proyecto', 'class="text" size="1"', $obj->coste_proyecto );
?></td>
```

El valor devuelto por esa función, una vez realizada la selección por parte del usuario y pulsar el botón “enviar”, es el índice del array asociativo contenido en $projects, que se corresponderá con el identificador del proyecto seleccionado.

- Para los campos de **edición de fechas** hemos “reutilizado” el código que hemos encontrado en el módulo projects. Para adaptar el código original al nuestro sólo fue necesario modificar el primer fragmento. En concreto usamos el siguiente código php, javascript y html:

```
// format dates
$df = $AppUI->getPref('SHDATEFORMAT');
$start_date = new CDate( $obj->tarifa_fecha_inicio );
$end_date = intval( $obj->tarifa_fecha_fin ) ?
            new CDate( $obj->tarifa_fecha_fin ) : null;
$style = (( $actual_end_date > $end_date) && !empty($end_date)) ?
            'style="color:red; font-weight:bold"' : '';
...
<link rel="stylesheet" type="text/css" media="all" href="<?php echo
DP_BASE_URL;?>/lib/calendar/calendar-dp.css" title="blue" />
<!-- import the calendar script -->
<script type="text/javascript" src="<?php echo
DP_BASE_URL;?>/lib/calendar/calendar.js"></script>
<!-- import the language module -->
<script type="text/javascript" src="<?php echo
DP_BASE_URL;?>/lib/calendar/lang/calendar-<?php echo $AppUI->user_locale; ?
>.js"></script>

<script language="javascript">
var calendarField = '';
var calWin = null;
```

```
function popCalendar( field ){
		calendarField = field;
		idate = eval( 'document.editFrm.project_' + field + '.value' );
		window.open(
		'index.php?m=public&a=calendar&dialog=1&callback=setCalendar&dat
		e=' + idate, 'calwin', 'width=280, height=250, scrollbars=no' );
}
/**
*		@param string Input date in the format YYYYMMDD
*		@param string Formatted date
*/
function setCalendar( idate, fdate ) {
		fld_date = eval( 'document.editFrm.project_' + calendarField );
		fld_fdate = eval( 'document.editFrm.' + calendarField );
		fld_date.value = idate;
		fld_fdate.value = fdate;
}
...
<td nowrap="nowrap">
<input type="hidden" name="project_start_date" value="<?php
		echo $start_date->format( FMT_TIMESTAMP_DATE );
?>" />
<input type="text" class="text" name="start_date" id="date1" value="<?php
		echo $start_date->format( $df );
?>" class="text" disabled="disabled" />
<a href="#" onClick="popCalendar( 'start_date', 'start_date');">
		<img src="./images/calendar.gif" width="24" height="12" alt="<?php
			echo $AppUI->_('Calendar');
		?>" border="0" />
</a>
</td>
```

Comentarios finales

Por último comentar que en algunos componentes no conseguimos escribir correctamente acentos y otros símbolos del Español en el entorno dotProject, como en algunas ventanas de confirmación y pestañas. La solución a esto se explica en el documento *Manual de programación en dotProject*.

VII. Manual de programación en dotProject

El presente manual describe algunas técnicas avanzadas para la creación de módulos para dotProject que no tenían cabida dentro del tutorial “proyectos SQA para dotProject”. Aquí mostraremos desde cómo obtener información hasta ejemplos de código de ciertas funcionalidades.

Dónde encontrar información

El mayor problema a la hora de encontrar documentación sobre cómo programar para dotProject es que esta información se encuentra actualmente distribuida entre dos wikis:

- http://sites.sakienvirotech.com/dp_docs/tiki/
- http://docs.dotproject.net

Toda la documentación se está trasladando a la segunda página, pero este proceso aún no se ha completado.

El segundo mayor problema es que la documentación disponible está dispersa y resulta difícil encontrar nuestras dudas en los foros. En cualquier caso son de visita aconsejada para estar al tanto de actualizaciones y de los problemas más habituales que puedan existir.

El tercer mayor problema es que la documentación técnica es muy escasa, limitándose a:

- Un **estándar de codificación** que se intenta seguir con muy escaso éxito. Dicen que programadores de módulos deberían aplicar un estándar similar a PEAR que se describe en esta página de la nueva wiki:
  http://docs.dotproject.net/index.php/Coding_Standard
  Pero lo cierto es que no se utiliza ni siquiera en el módulo Einstein que tienen a modo de tutorial.

- Un **modelo de permisos de la base de datos**, que se describe aquí:
  http://sites.sakienvirotech.com/dp_docs/tiki/tiki-index.php?page=Permissions+Theory+-+v2.0+and+above
  Para que los programas sean compatibles con las versiones anteriores a la 2.0 se deben usar las funciones que aparecen en el permissions.php que hay en el directorio /includes:
  http://sites.sakienvirotech.com/dp_docs/tiki/tiki-index.php?page=Coding+Modules+for+v2.0+Permissions

- Una explicación sobre cómo incluir variables a las **variables de sistema** que tiene dotProject por defecto. Para añadir opciones de configuración se recomienda añadir valores a la tabla 'config' de la base de datos como se explica aquí:
  http://sites.sakienvirotech.com/dp_docs/tiki/tiki-index.php?page=Config+Vars+and+Options

- Una explicación sobre cómo **realizar traducciones** de nuestro módulo a cualquier idioma como se explica aquí:
  http://sites.sakienvirotech.com/dp_docs/tiki/tiki-index.php?page=Creating+a+Translation
  Posteriormente detallaremos este apartado.

- Una descripción del **framework utilizado**. Lo han dividido en:
  - El sistema de ficheros, variables y métodos globales, y métodos de la clase 'Object':
    http://sites.sakienvirotech.com/dp_docs/tiki/tiki-index.php?page=dotProject+Framework
  - Una descripción de la arquitectura que puede ser consultada aquí:
    http://sites.sakienvirotech.com/dp_docs/tiki/tiki-index.php?page=dotProject+Architecture
  - También existe un esquema de la base de datos en pdf y xml aquí:
    http://sites.sakienvirotech.com/dp_docs/tiki/tiki-index.php?page=Database+Schema

Como vemos, no existe ninguna descripción detallada de las funciones que nos facilita la plataforma, ni los argumentos que utilizan, ni las variables de sistema que podemos utilizar. No pudiendo recurrir a la documentación “oficial”, nos quedarán las siguientes opciones:

- “Reutilizar código” de otros módulos existentes, limitándonos a renombrar variables e intuyendo la funcionalidad a partir de los escasos comentarios que encontremos.

- Consultar el código en el que están programadas esas variables/funciones, que se podrá encontrar en:
  - \classes
  - \functions
  - \includes
  - \install
  - \lib

  dependiendo del elemento de que se trate.

Programación de addons

La programación de módulos (addons) para dotProject se realiza a través de módulos programados en PHP. Estos módulos son fácilmente gestionados por el Administrador de módulos que nos permite instalar, desinstalar, ocultar o hacer visible cualquier módulo, ya sea original o addon. Cada nuevo módulo simplemente añade una pestaña a la GUI, pudiendo programar en ella cualquier nueva funcionalidad.

Si queremos modificar alguno de los módulos originales tenemos dos opciones:

- Modificarlo sin más, haciendo que nuestro trabajo sea incompatible con nuevos desarrollos.

- Reproducir la funcionalidad del módulo original copiando su funcionalidad y ocultándolo mediante el administrador de módulos (ya que no es posible desinstalar los módulos originales). El cómo realizar esta copia se explicará más adelante.

Sin embargo, la forma más habitual de realizar modificaciones a los módulos originales será añadiéndoles pestañas. Con esto no afectamos a la funcionalidad original del módulo en el que la integramos facilitando que nuestro trabajo sea compatible con futuras versiones de dotProject. La forma de añadir estas pestañas se explica más adelante.

Convenios de nombres

Todos los módulos de dotProject siguen un cierto convenio a la hora de asignar nombres a variables y valores de la base de datos. Las ventajas de seguir este convenio van desde la facilidad de comprensión y la portabilidad del código hasta la facilidad de deducir nombres de variables sin necesidad de recurrir a la documentación.

A continuación mostraremos algunos de ellos:

- Ficheros.
  - addobjeto.php o addedit.php para añadir o editar objetos.
  - view.php para ver el objeto más importante del módulo.
  - vw_pestaña.php para cada pestaña visible desde el index del módulo.
  - index.html que será el que defina cuándo y cómo ver cada pestaña.
  - do_módulo_aed.php para definir la interfaz con la base de datos.
  - módulo.class.php, se explica a continuacion.

- Clases
  - Se declaran en un fichero módulo.class.php, si bien en setup.php se encuentra la sentencia de creación de la tabla.
  - La clase de configuración/instalación/desinstalación en setup.php se llamará CSetupObjetos, respetando las mayúsculas y el nombre en plural.
  - La función de creación de módulo.class.php se llamará CObjetos(), respetando las mayúsculas y el nombre en plural.
  - Cada atributo definido en la clase Objetos se llamará objeto_atributo, siendo objeto_id su identificador principal. En caso de que el atributo haga referencia a otra tabla se escribirá como objeto_tabla, con el nombre de la tabla en singular.

- Tablas de la base de datos
  - Se sigue el mismo convenio que se acaba de describir para las clases: Nombre de tabla en plural y atributos con nombre de tabla en singular seguidos de guión bajo y nombre de atributo.

- Funciones javascript
  - Las funciones tipo Submit serán llamadas SubmitIt().
  - Las funciones tipo Delete serán llamadas DelIt().
  - Las funciones que abran ventanas tipo pop up se llamarán popNombreVentana().
  - Las funciones para asignar valores de vuelta de ventanas tipo pop up se llamarán setNombreVentana().

Sistema de traducciones

Como dotProject es una plataforma con una comunidad internacional de desarrolladores a sus espaldas, todo lo que hemos programado para ella se ha programado pensando en su eventual publicación. Por tanto tanto los textos mostrados por pantalla como las variables utilizadas se han programado en Inglés. Sin embargo todos los textos mostrados en pantalla estarán en Español gracias al fabuloso sistema de traducciones que nos facilita el entorno.

Para cada módulo que realicemos debemos crear su correspondiente fichero de traducción al Español. Como se indica en los ficheros de traducción, se debe usar obligatoriamente el sistema de traducciones del módulo de administración de dotProject, nunca editarlos a mano. Los ficheros de traducción se encuentran en el directorio /locales, con versiones en Inglés (/en), que es la que toman como referencia las demás traducciones, y en todos los idiomas que nos hayamos descargado (en principio, sólo Español (/es)).

El primer paso será programar el módulo utilizando la variable $AppUI cada vez que queramos que un texto sea traducido, dándole un valor por defecto en Inglés:

```
echo $AppUI->_('Description');
```

Este valor por defecto se traducirá automáticamente si está incluido entre las traducciones comunes de DotProject (fichero common.inc), cosa poco frecuente. Hay que tener en cuenta que los textos que introduzcamos son “case sensitive”.

En menús desplegables tenemos la opción de traducir los “elementos desplegados” poniendo a true el último argumento de la función arraySelect():

```
echo arraySelect($projects, 'requisite_project','size="1" class="text"',$obj->requisite_project,true);
```

Por último, en los mensajes incluidos dentro de código javascript deberemos usar como argumento UI_OUTPUT_JS:

```
alert( '<? echo $AppUI->_('Requisite needs a description', UI_OUTPUT_JS); ?>' );
```

De otro modo no se visualizarán ni eñes, ni acentos, ni otros símbolos propios de la lengua de Cervantes.

Una vez realizado el módulo en Inglés, entraremos en el administrador de traducciones (Sistema | Administración de traducciones) e introduciremos todos los términos que queramos que se traduzcan en idioma Inglés, seleccionando módulo e idioma con los menús desplegables de la parte superior derecha de la pantalla. A continuación seleccionaremos Español en el mismo desplegable y asociaremos a cada texto su correspondiente traducción.

A la hora de distribuir el módulo habrá que tener en cuenta que la traducción se encuentra en /locales/es/modulo.inc y /locales/en/modulo.inc, es decir, fuera del directorio del módulo que hayamos realizado.

Finalmente, decir que se ha encontrado un error en el sistema de traducciones de dotProject. Si el módulo traducido tiene componentes en otros módulos (por ejemplo pestañas) los términos no se traducirán salvo que existan también dentro de las traducciones comunes (fichero common.inc) o en las del módulo en el que hayamos insertado nuestra pestaña.

Técnicas de programación

A continuación se muestra una colección de técnicas para realizar alguna de las modificaciones más frecuentes a nuestros módulos. A modo de índice, las técnicas que se explican son:

- Cómo ver el contenido y la estructura de una variable en PHP
- Cómo cambiar el directorio de un módulo
- Cómo añadir botones de borrado
- Cómo añadir menús desplegables
- Cómo ejecutar sentencias MySQL
- Cómo añadir atributos a la base de datos
- Cómo añadir tooltips
- Cómo añadir pestañas a otros módulos
- Cómo crear pop-ups
- Cómo crear tablas ordenables
- Cómo realizar una copia de uno de los módulos originales
- Cómo utilizar los “valores del sistema”

Pasamos a explicar cada uno de ellos:

- **Cómo ver el contenido y estructura de una en PHP:**

  El lenguaje PHP nos facilita una función mucho más completa que *echo*. Es:

  ```
  print_r($variable);
  ```

  y nos muestra el contenido y la estructura interna de la variable que le pasemos como parámetro. Esto es especialmente útil si tienes poca experiencia con este lenguaje, ya que al ser un lenguaje no tipado es bastante difícil saber qué contiene exactamente cada variable.

- **Cómo cambiar el directorio de un módulo**

  Esto sólo es aplicable a los módulos tipo addon, es decir, los que no vienen con la distribución original de dotProject. Para que todo funcione correctamente basta con coger uno de estos módulos hacer lo siguiente:

  1. Cambiar en el setup.php el contenido de $config['mod_directory'].
  2. Renombrar todos los “m=modulo_proyecto” con el nombre del nuevo directorio.

- **Cómo añadir botones de borrado**

  Para agregar un botón de borrar a través de un icono junto a cada elemento del un listado de elementos, como costs en este caso, se debe:

  1. Añadir en la tabla de view_costs.php:

     ```
     if ($canDelete) {
         ?>
         <a href="javascript:delMe(<?php echo $row["cost_id"];?>);">
         <?php echo dPshowImage( './images/icons/stock_delete-16.png', 16, 16, '' ); ?>
         </a>
         <?php
     }
     ```

     Como vemos, hemos utilizado la imagen que se encuentra en el directorio /images/icons y la mostramos utilizando la función que dotProject nos da para ello, dPshowImage(). La variable $canDelete es una variable global que se importa en algún lugar al principio del código de la página y nos dice si el usuario que la abre tiene privilegios de edición/borrado sobre el objeto.

  2. Añadir esto en alguna parte del fichero:

     ```
     <script language="javascript">
     function delMe( x ) {
         if (confirm( "<?php echo $AppUI->_('Are you sure?', UI_OUTPUT_JS);?>" )) {
             document.frmDelete.cost_id.value = x;
             document.frmDelete.submit();
         }
      }
     </script>
     <form name="frmDelete" action="./index.php?m=costs" method="post">
     <input type="hidden" name="dosql" value="do_cost_aed" />
     <input type="hidden" name="del" value="1" />
     <input type="hidden" name="cost_id" value="0" />
     </form>
     ```

     El fragmento del script es la función en Javascript que atenderá la llamada vinculada al click sobre el icono de borrado. Aquí podremos incluir otras instrucciones que comprueben que los campos tengan valores correctos. Esto es especialmente útil en las vistas de creación de objetos, ya que MySQL por si solo no realiza un control de errores adecuado. Además nos permitirá asignar valores a determinados campos sin necesidad de recargar la página.

Por otro lado, el formulario “frmDelete” contiene la información que necesita nuestra interfaz con la base de datos (do_cost_aed en este caso) para aplicar los cambios solicitados sobre ella. Como se puede ver, todas las inputs que aparezcan se declararán como ocultas. Sus valores son los tres que aparecen para que la interfaz los interprete correctamente.

- **Cómo añadir menús desplegables**

Esto es tan sencillo como utilizar la función arraySelect que nos proporciona DotProject. Para ello necesitamos un array indexado, en el que el valor indexado es el que verá el usuario en la lista y el valor que indexa es el que el navegador asignará al elemento que seleccione.

Para el caso de la selección de roles de usuario utilizaremos para almacenar esta lista la variables $projects:

```
require_once( $AppUI->getModuleClass ('projects' ) );
$aux = new CProject();
$projects = $aux->getAllowedRecords( $AppUI->user_id, 'project_id,project_name', 'project_name' );
```

Vemos que la clase CProject tiene una función específica para cargar todos los nombres de proyecto asociados directamente a su ID, usando la función getAllowedRecords. Esta función tiene la ventaja de que se hace un control de permisos que devuelve únicamente los valores que puede visualizar el usuario que abre la página. Sin embargo esta función no siempre existirá por lo que tendremos que crear la lista usando una consulta MySQL.

Si queremos que haya un espacio en blanco al principio de la lista para que el usuario pueda dejar el desplegable sin ningún valor, deberemos añadir esta instrucción, en el que asignamos el valor cero a una cadena de caracteres vacía, pero se puede poner cualquier otro par de valores:

```
$projects = arrayMerge( array( '0'=>'' ), $projects );
```

Por último recomiendo encarecidamente utilizar también este fragmento de código después de cargar los valores:

```
foreach ($projects as $id => $value) {
    if (strlen($projects[$id]) > 20) {
        $projects[$id] = substr( $projects[$id] , 0 , 18)."...";
    }
}
```

Con él limitaremos la longitud máxima de los elementos del desplegable, ya que será frecuente que los valores de este campo tengan longitudes variables que podrán hacer que las proporciones que hemos dado a las casillas de nuestras tablas se vayan al garete. En este caso, si algún valor tiene más de 20 caracteres se trunca utilizando la función substr y se añaden puntos suspensivos para que el usuario sepa que se trata de un texto truncado.

Para terminar, insertaremos el arraySelect en alguna parte de nuestra página, utilizando la instrucción:

```
<td ><?php
    echo arraySelect($projects, 'filter_project', 'size="1" class="text" ', $filter_project
    , false);
?></td>
```

Esto funcionará de maravilla, pero si además queremos que cuando seleccionemos algún valor tenga lugar alguna acción definida mediante una acción en Javascript, añadiremos a los parámetros size y class el valor onchange=”función()”. Para verlo más claramente:

```
<td ><?php
    echo arraySelect($projects, 'nombre_campo',
        'onchange="funciónJavascript();" size="1" class="text" ',
    $nombre_campo_inicial , false);
?></td>
```

Esto es útil por ejemplo para seleccionar una tarea que pertenece a un determinado proyecto y queremos que al seleccionar el proyecto se restrinja la lista completa de tareas a las del proyecto que seleccionemos.

- **Cómo ejecutar sentencias MySQL**

El entorno dotProject nos facilita multitud de instrucciones de acceso a la base de datos. La más general, que servirá para todo tipo de operaciones es db_exec(). Aquí se muestra un ejemplo en el que se borran entradas de la tabla “requisite_associated_companies” que tengan como requisito asociado un determinado identificador de requisito:

```
$sql = "DELETE FROM requisite_associated_companies".
    " WHERE rac_requisite=".$requisite_id;
db_exec( $sql );
db_error();
```

Es aconsejable separar la cadena en líneas distintas dentro del código PHP para facilitar su comprensión.

Por otro lado, también estaremos interesados en recuperar información de la base de datos. Para ello existe la función db_loadList que se usa como se muestra a continuación:

```
$sql = "SELECT contact_first_name, contact_last_name, wage_amount".
    "FROM contacts, wages, users ".
    "WHERE wage_user = user_id AND user_contact = contact_id “;
$wages = db_loadList( $sql );       // retrieve a list (in form of an indexed array) via an abstract db method
```

$wages será un array indexado con el contenido de la consulta. Aquí es muy recomendable utilizar phpMyAdmin para comprobar que la consulta devuelve el valor que esperamos y la función print_r para comprobar que $wages también contiene los valores que queríamos.

- **Cómo añadir atributos a la base de datos**

  Aunque podemos añadir los atributos a través de operaciones del tipo “alter table” en MySQL siguiendo el procedimiento explicado en el punto anterior, es mucho más cómodo cargar los valores que necesitaremos en nuestros objetos al instalar el módulo. Para ello tendremos que añadir los nuevos atributos en los ficheros siguientes:

  - en setup.php, hay que añadir el atributo a la consulta de creación de tablas en MySQL dentro de la función install.

  - en sqa_proyectos.class.php, se debe añadir una variable con valor a null como atributo de la clase que hará de interfaz con la base de datos.

  - en vw_listado ponerle una columna si queremos visualizarlo.

  - en addedit poner recuadro correspondiente para añadir ese nuevo atributo.

  El siguiente paso será reinstalar el módulo para que los cambios tengan efecto. Recordamos que es conveniente seguir el convenio de nombrado explicado en otra sección de este mismo documento.

- **Cómo añadir tooltips**

  Los tooltips son un tipo especial de link que incluye una función para abrir el tooltip (onmouseover) y otra para cerrarlo (onmouseout). El tooltip en cuestión se muestra con la función overlib. Como la explicación de su funcionamiento es algo compleja, recomiendo utilizarlo tal cual se muestra aquí:

```
<a href="?m=requisites&a=addrequisite&requisite_id=<?php
    echo $row["requisite_id"]
?>" onmouseover="return overlib( '<?php
    echo "Esto es un tooltip"
?>' );" onmouseout="nd();"><?php
    echo htmlspecialchars( $row["requisite_name"], ENT_QUOTES );
?></a>
```

Se trata de un link a la vista de edición de un requisito mostrado en un listado. Al pasar el ratón por encima se muestra el mensaje “Esto es un tooltip”, texto que puede ser cambiado por uno que dependa del contenido del requisito. La tercera línea muestra el nombre del requisito sin más.

- **Cómo añadir pestañas a otros módulos**

DotProject nos permite integrar pestañas dentro de otros módulos que estén preparados para ello. Personalmente sólo he conseguido insertar pestañas en los lugares que describo aquí, pero no se si será posible hacerlo en otros.

El procedimiento es tan sencillo como incluir dentro del directorio de nuestro módulo un fichero con este nombre:

módulodestino_tab.nuestroMódulo.php

O bien, dependiendo del módulo:

módulodestino_tab.otrosficheros.nuestroMódulo.php

A continuación eliminaremos las cookies y volveremos a entrar a dotProject para que el navegador olvide la configuración de marcos que tenía. Y ya está.

Los lugares en los que he conseguido añadir pestañas, y el nombre de fichero que deben tener son:

projects_tab.costs.php	para el index del módulo projects
companies_tab.costs.php	para el index del módulo companies
departments_tab.view.costs.php	para el view del módulo departments
tasks_tab.view.costs.php	para el view del módulo tasks
admin_tab.viewuser.costs.php	para el view del módulo de administración de usuarios

- **Cómo crear pop-ups**

Vamos a explicar cómo crear una ventana independiente se abra, seleccionemos un valor, y que retorne a donde nos encontrásemos asignando un valor nuevo a algún campo del formulario desde el que lo lanzamos.

1. En primer lugar, ponemos algún evento en el formulario que nos ejecute una función Javascript dentro del mismo formulario, como por ejemplo “popTasks()”, que permitirá elegir una tarea haciendo click en alguna de las filas del listado que se mostrará:

```
<input type="hidden" name="requisite_task" value="<?php
    echo $obj->requisite_task;
?>
<input class="button" type="button" name="task_selector" value="<?php
    echo $AppUI->_('Search');
?>" onClick="popTasks('task_selector');" />
```

   Le hemos pasado el nombre del campo desde el que lo lanzamos porque habrá ocasiones en las que querramos dar distinto comportamiento a la función dependiendo del lugar desde el que se llame. En este ejemplo será irrelevante.
   Vemos que también hemos creado un atributo oculto llamado requisite_task, que será donde almacenemos el valor devuelto por la ventana. Este campo se inicializa con el valor que tuviera antes de cargar el formulario, y será el que almacene el valor que queramos pasar a la base de datos.

2. A continuación creamos la función popTasks dentro del código del mismo fichero en el apartado correspondiente a javascript:

```
function popTasks ( field ) {
    var f = document.editFrm;
    window.open('./index.php?m=public&a=task_selector&dialog=1&call_back=s
    etTask'+
                '&selected_task_id='+f.requisite_task.value,
                'tasks',
                'height=650,width=550,resizable,scrollbars=yes');
}
```

   Vamos a utilizar la función window.open para abrir la página que contendrá la ventana que queremos desplegar. Los tres argumentos que se pasan a esta función son:

   - La url del lugar donde se encuentra.
   - El nombre de la tabla a la que vamos a acceder.
   - Las dimensiones y forma de abrir la ventana.

Dentro de los valores de la url distinguimos entre los siguientes argumentos (separados por el símbolo “&”):

- m=public, que es el módulo en el que se encontrará el fichero con el código de la ventana que vamos a abrir. Lo hemos incluido en el directorio /public para seguir la metodología de dotProject a la hora de crear este tipo de ventanas, supuestamente para que el código sea utilizable por otros módulos.

- a=task_selector, que es el fichero dentro de /public en el que se encuentra el código de la ventana, en este caso task_selector.php

- dialog=1, es para mostrar la ventana de forma independiente. De otra forma se abriría en la misma ventana que la abre.

- call_back=setTask, se utiliza para especificar la función Javascript que atenderá el resultado de la ventana. Se describirá a continuación.

- '&selected_task_id='+f.requisite_task.value, se utiliza para pasar como argumento a la nueva ventana el valor que hubiera en el campo que queremos modificar, que en este caso es requisite_task.

3. El siguiente paso es programar el código de la ventana que se abrirá, que como ya hemos dicho estará en el fichero task_selector.php dentro de /public. En el código debe haber:

   - Una elemento que lance una función javascript, como por ejemplo este link que lanza la función setTaskID con el identificador de la tarea seleccionada en una lista ($row[“task_id”])

```
<a href="#" onclick="setTaskID( <?php
    echo $row["task_id"];
?> )">
```

   - La función a la que se accederá al hacer click sobre la tarea $row[“task_id”] tendrá este aspecto:

```
<script language="javascript">
    function setTaskID (task) {
    window.opener.<?php echo $call_back;?>(task);
```

```
        window.close();
    }
    </script>
```

Vemos que el argumento que le pasamos será incluido en la llamada de vuelta mediante la función window.opener. El siguiente paso es cerrar la ventana con window.close. Lógicamente, en la cabecera del código habremos cargado previamente las variables que hemos pasado dentro de la url:

```
$call_back = dPgetParam($_GET, 'call_back', null);
$selected_task_id = dPgetParam($_GET, 'selected_task_id', 0);
```

- Si queremos añadir un botón que cierre la ventana sin modificar ningún valor podemos usar un botón de cerrar como este:

```
<input class="button" type="button" name="cancel" value="<?php
    echo $AppUI->_('Close');
?>" onClick="window.close()" />
```

4. Para terminar creamos la función setTask en el formulario original, que iba a ser la que capturase el valor devuelto por el pop up. En este caso el efecto que queremos que produzca es que asigne el valor devuelto al campo oculto requisite_task:

```
function setTask (task) {
    var f = document.editFrm;
    f.requisite_task.value = task;
}
```

- **Cómo crear tablas ordenables**

La técnica que vamos a utilizar para realizar tablas ordenables consiste en poner un link en todas las cabeceras de la tabla que recargue la página con un parámetro que indique por qué atributo queremos que se ordene. Para ello es necesario que el listado se muestre a través de una consulta MySQL como la siguiente:

```
$q = new DBQuery;
$q->addQuery('requisite_code');
...
$q->addQuery('project_name');
$q->addTable('requisites');
$q->addTable('projects');
$q->addWhere('requisite_project=project_id');
$q->addOrder( $order );
$requisites = $q->loadList();
```

En este caso hemos utilizado la clase DBQuery, pero sería equivalente a utilizar db_loadList() con la consulta pasada como una cadena de caracteres. La variable $order contendrá nombre del argumento sobre el que ordenaremos el listado, que en este caso podrá valer requisite_code o project_name. Para cargar el valor de esta variable, que habrá sido pasada como un parámetro a través de la url usamos la función dPgetParam():

```
$order = dPgetParam( $_GET, "orderby", "requisite_code" );
```

Para terminar añadimos un link a las cabeceras de las tablas con el argumento orderby seguido del atributo por el que se filtrará cuando se haga click en él:

```
<a href="?m=requisites&orderby=requisite_code" class="hdr"><?php
        echo $AppUI->_( 'Code' );
?></a>
<a href="?m=requisites&orderby=project_name" class="hdr"><?php
    echo $AppUI->_( 'Project' );
?></a>
```

- **Cómo realizar una copia de uno de los módulos originales**

Es posible que en algún momento podamos necesitar realizar modificaciones sobre alguno de los módulos que trae dotProject por defecto, como por ejemplo tener una versión del módulo Projects adaptada a cada departamento de la empresa. La forma de hacerlo requiere un renombrado sistemático de variables entre otras cosas como se muestra a continuación:

1 Copiar el contenido del módulo a un nuevo directorio.

2 Eliminar los siguientes ficheros de nuestra copia, pues no son necesarios ya que el módulo original conservará su funcionalidad:
CMóduloOriginal.class.php
do_MóduloOriginal_aed.php
OtrosMódulos_tab.*.php

3 Crear un setup.php como si se tratara de un módulo nuevo. Los módulos originales no lo tienen, así que habrá que tomar como referencia algún módulo de ejemplo. Por tanto habrá que asignar valores a cada variable de configuración. Por cierto, es muy importante poner mod_type = user o las consecuencias serán catastróficas.

4 Realizar los siguientes cambios en todos los ficheros del módulo:

4.1 Si aparece alguna instrucción del tipo CMóduloOriginal() en alguna parte del código deberemos importar el módulo. Para ello utilizamos la instrucción require_once:
require_once($AppUI->getModuleClass('móduloOriginal'));
Esto nos dará acceso a las funciones que se encuentren en el fichero móduloOriginal.class.php.

4.2 Renombrar con el valor de nuestro módulo cada:
...m=modulo_proyecto...
.../directorio módulo/'...
Con esto utilizaremos los ficheros de nuestro módulo en lugar de los del original.

4.3 Si aparece algún arraySelect($listaValores, ...) que no se cargue correctamente añadir antes:
$listaValores = dPgetSysVal('Lista');
Donde 'Lista' es una variable definida dentro de la clase del módulo original que

contiene todos los valores posibles para ese campo. La función dPgetSysVal() nos importa todos los valores de la lista.

5 Por último, realizar los siguientes cambios en la GUI:

5.1 Cambiar el título del módulo en todas las páginas en las que apareciera (CtitleBlock)

5.2 Opcionalmente, copiar en /images un icono propio y modificar esta línea en cada view.php o vw_*.php:

$titleBlock = new CtitleBlock(... 'nuestro logo.gif' ...);

5.3 También opcionalmente, desactivar el módulo original utilizando el administrador de módulos.

A partir de este momento podremos modificar cualquier elemento de nuestro módulo sin afectar a los que trae dotProject inicialmente, de modo que nuestro módulo no se vea afectado en el caso de una eventual actualización de la plataforma.

La única desventaja que tiene esta técnica, es que no es posible añadir pestañas desde otros módulos. Mientras que para añadir una pestaña en uno de los módulos originales bastaba con poner:

módulodestino_tab.nuestroMódulo.php

esto no es aplicable a la copia que hemos realizado, por lo que cualquier pestaña que queramos incluir deberá programarse dentro del propio módulo copiado.

- **Cómo utilizar los “valores del sistema”**

  DotProject tiene una funcionalidad llamada “valores del sistema” por la que es posible definir ciertas variables por parte del usuario a través de una opción del menú de administración. En nuestro caso estábamos interesados en definir una variable llamada RequisiteStatus que se creara en el momento de instalar el módulo Requisites y que pudiera ser modificada por el usuario en cualquier momento. En concreto contendría esta asociación de valores:

  0 - Propuesto
  1 - Aprobado
  2 - En desarrollo
  3 - Finalizado
  4 - Validado
  5 - Facturado

  Para manipular este tipo de variables, dotProject utiliza una tabla en la base de datos llamada “sysvals”, lo que nos da control absoluto desde nuestro código sobre el contenido que pueda tener cualquier variable. En nuestro, para la creación de esta variable utilizamos la consulta:

```
$sql = "INSERT INTO sysvals ".
            "(sysval_key_id, sysval_title, sysval_value) ".
        "VALUES ".
            "(1, \"RequisiteStatus\", \"".
                "0|Propuesto\n".
                "1|Aprobado\n".
                "2|En desarrollo\n".
                "3|Finalizado\n".
                "4|Validado\n".
                "5|Facturado\")";
db_exec( $sql ); db_error();
```

  Al tratarse de valores que aparecerán en menús desplegables, el formato que deben tener debe ser el mostrado, es decir, pares identificador, nombre separados por el caracter “|”, con cada entrada en una línea distinta.

  Incluiremos esta sentencia en la función install() del fichero setup.php para que la variable se cree cada vez que se instale el módulo, y con la siguiente sentencia, dentro

de la función remove():

```
$sql = "DELETE FROM sysvals ".
        "WHERE sysval_title = \"RequisiteStatus\"";
db_exec( $sql ); db_error();
```

eliminaremos la variable cuando se desinstale.

Cada vez que queramos utilizar una de estas variables en nuestro código, en primer lugar la importaremos con:

```
$status = dPgetSysVal( 'RequisiteStatus' );
```

Y la utilizaremos en nuestros arraySelects así:

```
<td width="33%" nowrap="nowrap"><?php
    echo arraySelect($status, 'requisite_status', 'size="1" class="text"', $obj->requisite_status , false);
?></td>
```