Universidad de Murcia

Facultad de Informática

Trabajo Fin de Grado

CARE de soporte a PANGEA basado en CMS:

PANTALASA-CMS

Autor: Juan Carlos Martínez Expósito

Juancarlos.martinez4@um.es

Tutor: Joaquín Nicolás Ros

Convocatoria de Septiembre 2013

Índice

Índice 3
Índice de figuras 5
Índice de tablas 6
1 Resumen 7
2 Extended abstract 8
3 Introducción y referencias históricas 14
3.1.1 Métodos SIREN y PANGEA 14
3.1.2 Actividades y tareas 16
3.1.3 Modelo de referencia de requisitos 23
3.1.4 Otras aproximaciones al soporte de PANGEA 28
4 Análisis de Objetivos y Metodología 29
4.1.1 Objetivo del trabajo 29
4.1.2 Metodología 29
4.1.3 Herramientas y servicios utilizados 29
5 Diseño y resolución 31
5.1 Especificación del problema 31
5.2 Análisis y selección de alternativas 31
5.2.1 Requisitos de la herramienta PANTALASA-CMS 32
5.2.2 Wordpress 32
5.2.3 Joomla 33
5.2.4 Drupal 33
5.2.5 Magnolia-CMS 34
5.2.6 Tabla comparativa 35
5.2.7 Conclusiones 35
5.3 Diseño 37
5.3.1 Modelo de Casos de Uso 37
5.3.2 Módulo PANTALASA_CMS 55
5.3.3 Módulos adicionales para Drupal 71
5.4 Implementación del prototipo 72
5.4.1 Vocabularios y taxonomías 72

5.4.2 Tipos de contenido 75
5.4.3 Menú de PANTALASA_CMS 104
5.4.4 Roles 110
5.4.5 Menús 114
5.4.6 Vistas 117
5.4.7 Formularios 133
5.4.8 Bloques 136
5.4.9 Otros 136
5.5 Validación 141
6 Conclusiones y vías futuras 160
6.1 Conclusiones 160
6.2 Vías futuras 161
7 Referencias 162
8 Anexos 164
8.1 Instalación 164
8.2 XML Schema de un documento de requisitos 166
8.3 XML Schema de un patrón 171

Índice de figuras

Figura 1 - Modelo en espiral del proceso básico de IR 15
Figura 2 - Modelo de proceso SIREN 15
Figura 3 - Modelo de proceso PANGEA 16
Figura 4 - Jerarquía de los roles en los grupos de trabajo 22
Figura 5 - Diagrama de estados de un requisito 25
Figura 6 - Jerarquía de plantillas de documentos en SIREN 28
Figura 7 - Casos de uso del responsable del repositorio 38
Figura 8 - Casos de uso del coordinador 39
Figura 9 - Casos de uso del moderador 39
Figura 10 - Casos de uso del representante de equipo 40
Figura 11 - Casos de uso del analista 40
Figura 12 - Casos de uso del usuario y usuario clave 40
Figura 13 - Esquema PANTALSA-CMS (simple) 57
Figura 14 - Esquema PANTALSA-CMS (completo) 58
Figura 15 - Diagrama conceptual 59
Figura 16 - Esquema de Drupal 64
Figura 17 - PANTALASA-CMS como módulo 65
Figura 18 - Pantalasa CMS en Drupal 65
Figura 19 - Estructura de direcorios de Pantalasa CMS 66
Figura 20 - Módulos necesarios para Pantalasa CMS 71

Índice de tablas

Tabla 1. PANGEA: Descripción de actividades I 19
Tabla 2. PANGEA: Entradas y salidas de actividades 20
Tabla 3. PANGEA: Roles 20
Tabla 4 - PANGEA: Responsabilidades de los roles I 21
Tabla 5 - PANGEA: Responsabilidades de los roles II 22
Tabla 6 - Tareas y porcentaje de tiempo 29
Tabla 7 - Herramientas y servicios usados 30
Tabla 8 - Tabla comparativa de CMS 35
Tabla 9 - CDU 1 - Gestionar catálogos 41
Tabla 10 - CDU 2 - Gestionar tipos de parámetros 42
Tabla 11 - CDU 3 - Gestionar proyectos 43
Tabla 12 - CDU 4 - Gestionar grupos de trabajo 44
Tabla 13 - CDU5 - Exportar documento de requisitos 44
Tabla 14 - CDU 6 - Gestionar usuarios de un proyecto 45
Tabla 15 - CDU 7 - Crear versión base de documento 46
Tabla 16 - CDU 8 - Gestionar documentos de requisitos 47
Tabla 17 - CDU 9 - Gestionar hilos de disución 48
Tabla 18 - CDU 10 - Gestionar requisitos 50
Tabla 19 - CDU 11 - Gestionar secciones de documentos 51
Tabla 20 - CDU 12 - Participar en hilos de discusión 52
Tabla 21 - CDU 13 - Usar un catálogo de requisitos 53
Tabla 22 - CDU 14 - Crear parámetro 54
Tabla 23 - CDU 15 - Consultar un proyecto 54
Tabla 24 - Tipos de ficheros en directorios 67
Tabla 25 - Información fichero pantalasa_cms.info 67
Tabla 27 - Tabla resumen de vocabularios 74
Tabla 28 - Resumen términos de vocabularios 75
Tabla 31 - Resumen de tipos de contenido y campos 104
Tabla 32 - Opciones de menú y navegación 110
Tabla 26 - Resumen de roles y permisos de Drupal usado 113
Tabla 33 - Formularios y sus funciones 136
Tabla 34 - Funciones de db_functions.inc 138
Tabla 35 - Funciones en common.inc 138
Tabla 34 - Funciones de content.inc 139

1 Resumen

Con este trabajo se pretende crear un prototipo funcional de PANTALASA, una herramienta de soporte para un repositorio de requisitos reutilizables basado en la metodología PANGEA, haciendo uso de un sistema de administración de contenidos web o CMS (*Content Management System*) de libre distribución. PANGEA es un método de ingeniería de requisitos adaptado al desarrollo global de software, basado en reutilización y que deriva del método SIREN. La implementación de PANGEA se extiende en este trabajo fin de grado con soporte para patrones de requisitos y folksonomías. En este trabajo se ha seleccionado el CMS Drupal en su versión 7 aprovechando su amplia funcionalidad y expansión a través de módulos para la implementación de la herramienta. Drupal permite crear vistas personalizadas para distintos tipos de usuario añadiendo filtros y argumentos a estas para hacerlas más potentes. También permite establecer referencias entre los distintos tipos de contenido para implementar las distintas relaciones entre los objetos del dominio que tengan representación en la herramienta. Permite una control muy minucioso sobre el acceso a tanto a vistas como a opciones de navegación y funcionalidad implementada así como hacerlo de una forma elegante y limpia. La funcionalidad de la herramienta se ha probado con un catálogo de internacionalización previamente disponible.

2 Extended abstract

In today's world, there was a lot of software development companies which treat of make quality systems.

Sometimes, client or users of a software product and the company which make product, lie in different countries. This may be a serious problem if isn't treated appropriately. One of problems is the time difference between company and client. Maybe happen that while the company's employees are working, the contact people on the client side are sleeping because is too late. This may cause that the people of both side, client and company, never will have a real time conversation. One of reasons of fail is the poor communication between parts.

Other problem are cultural differences. Maybe happen that use a certain color or write certain expression hold different meanings for the client that for the development company.

Also, may happen that the client needs some product's part too soon and the company's employees should work more hours and law don’t allow it.

The solution that many companies choose is distribute work in different development companies or different offices of the same company around the world. This allows, for example, become aware of client's culture, improve communication between client and company or increase development time per day.

All this has led to emergence a new concept, Global Software Development (GSD). Consist in geographically and temporally company's resources distribution to improve the software development and improve the final result.

To create a quality software product, is needed a development process with several phases and sequential or iterative stages through which the product is developed. Developing a software product, is not an easy task since we are talking of something intangible which, sometimes, we have not an overview until we are in the final stages of its development. But it may happen that when you get to the latter stages we realize some kind of error or a feature that is not satisfied. The cost of solving an error increases exponentially as it becomes advance in the development process so that increase costs and require more time to develop because the error must be corrected and undo what has been done. This implies violate the contracts with our customers and that our product will be cancelled.

For these reasons, is very important to understand what the customer wants and what our product should offer before moving forward with the development. This is task of requirements engineering. We can say that requirements engineering covers all tasks related to determination of the conditions that should be met a software product and needs that should be satisfied. All these conditions and requirements are collected from the different project stakeholders. Sometimes these conditions and needs may be in conflict and should be try to settle.

One of the main objectives of requirements engineering is to obtain a collection of requirements derived from the needs and conditions of the product before advancing optimal design phases.

SIREN is a requirements engineering method based on requirements reuse. Some requirements used for a particular software product, can be abstracted to generate generic requirements that may apply to other software products.

These generic requirements can be grouped and collected according to a given domain or profile and stored for later reuse in a specific product. These requirements catalogs are refined over time so that every time you have an enhanced set of requirements.

But SIREN is not enough if the software development lies in a distributed environment. We need an adapted method for requirements engineering in GSD.

PANGEA is an extension of SIREN method for global software development. This method, PANGEA, improves or extends SIREN for adapt it to a distributed environment where the software development is carried out in different countries, cultures, etc.

PANGEA method is a complex requirement engineering method which has several phases and each phase has several task. Each task is performed by a user role and is necessary a communication between this user roles.

Perform each one of this tasks is complex or not easy so we need a tool that helps developers to implement PANGEA on theirs software development process.

The aim of this work is develop a tool prototype that supports PANGEA based in CMS. The main aspects to be taken into account in the tool's development have been the maintenance of a catalog requirements repository and reuse all or part of these requirements, concurrency management, users management, requirements management and powerful search in catalogs.

CMS is a content manager system. The main feature of a CMS is that allow you to create a functional website without programming knowledge. Usually, a CMS is maintained by a developers community. This developers community implements common functionality like users management, displays, user views, etc. The CMS users only need create his own contents types and configuration. The CMS does the rest. Also, the users don't worry about the CMS maintenance because developers community, periodically, releases an update or a bug correction which install automatically.

The CMS installation, usually, is very simple and fast. The installation process is guided through a wizard that allow you to set the main features of your site. Other CMS's advantage is that is very easy edit a website and create, add or improve some functionality.

The most CMS allow you create a website with his own functionality implemented already. But, if you need something more complex or custom, a lot of CMS allow create your own functionality and add it to CMS through its API.

Today, the most used CMS in the world are free and are based in PHP. For example: Wordpress, Joomla or Drupal.

The CMS must support features such as being able to work in a distributed environment where transactions are executed simultaneously so must provide concurrency management. Another aspect to consider is the parameterization of requirements where requirements have a parameter that can take a set of values of a particular type. The meta information associated to a requirement is also important in PANGEA. Selected CMS should allow join this meta information to a requirement.

Additionally, requirements may be related by traces that can be of various sorts and should be taken into account when reusing a requirement because it can depend on others.

To facilitate reuse existing requirements at repository, CMS should provide a powerful search engine which allows apply filters for requirement's attributes as well as for catalog where we want to search.

Also, chosen CMS should provides a based role access control for tool functionality. We must differentiate between a user and other which have same role but belong to different projects.

Other very important feature of chosen CMS is allow edit and expand tool functionality in the future for other aspects of PANGEA methodology.

After a study of more used CMS today, chosen CMS was Drupal version 7.

Drupal meets all design constraints imposed, allowing, at the expense of sacrificing the ease of implementation, implement all required functionality in this prototype. Perhaps, the difficulty in development such a tool is the main disadvantage found in Drupal but are overtaken by the powerful functionality that allows develop modular way and tidy.

The process that was followed to develop the tool was started studying PANGEA method, mainly, reuse requirements. Subsequently he has made a use case based analysis where have picked different interactions between actors and the system and the actions that the system performs.

Once we know clear the functionality that must be implemented we need to study Drupal API. Because of the complexity of this tool has not been able to make direct use of the functionality already implemented by Drupal and so we have had to develop the tool with a module. This way of implementing the tool makes it very easy to install with little user intervention. It also makes it easy to you install the tool on any server that supports Drupal’s requirements. Another advantage of implementing the tool like a module is the ability to extend the functionality of the tool with other existing modules or created expressly for the tool.

One aspect that Drupal must note because it is essential for the implementation of the tool is the Drupal hook. The hook is a known set functions by Drupal that allow Drupal interact with our module and enlarge functionality including navigation menus, content blocks as well as own module installation and uninstallation into the system.

Drupal, mainly works with PHP functions and PHP array. Before to start to coding the tool we have needed study the PHP manual related with arrays and how PHP works with them. Also is very important know the functions that PHP provides to manage array and arrays index.

Also is very important to have certain PHP knowledge for manipulate variables and functions as well as have access to PHP documentation and manual.

The scheme has been followed for the implementation begins to create different types of content and their relationships, create different user roles and set up basic permission, define navigation options and functionality available and incorporate user views. These user views has been developed with the module "Views" which considerably facilitates the creation of views and allowing to create it visually and allows to export his PHP code and store it in a directory of your module and then include it in the tool at installation time.

For this implementation have been used several modules developed by the Drupal community where some of them are included in the distribution itself Drupal and others are available for download and installation. Defining relationships between modules streamlines the installation process since Drupal resolve dependencies.

Access to user views and functionality and navigation options implemented has been implemented manually in order to have better control over what actions can run or what views a particular user can access.

Access to navigation and operations options has been implemented using Drupal API allowing you to implement this access control in a certain code function meanwhile user views access control has been implemented following the module API "views", which has been used to create user views.

For restricted values for certain content types attributes has been used the "taxonomy" module. That's module allow to create vocabularies and terms of these vocabularies.

For users and content types as well as role creation, we have used modules that offers Drupal into his core distribution.

At the installation time we have encountered a problem. There are user views that references other user view and we can't know which user view will be create first. For this, the system administrator must check a little set users views and check that the references has been established correctly.

The same problem appears with term’s vocabulary references in view and user references in views and the system administrator must check it.

The tool code has been organized in few directories based in the code functionality. Drupal has been developed in PHP code so all tool code was written in PHP code.

As database manager has been chosen MySQL because is a free database manager y very used in a lot of hosting services. This is not very important because Drupal has a database API that allow you to abstract from the used database manager.

Other tool aspect is that allow to export requirements documents where sections and requirements are including. The chosen format to export documents has been XML because it's a standard. This XML files are based in a XML Schema which is included with de tool. The XML Schema file defines how a requirements document should be represented within tool.

Also, we use XML to allow users use requirements patterns in the tool. That allow users create his own requirement pattern and add it to the tool. This requirements patterns will be available into the system. Five requirements patterns has been included into the tool, (Mavin's patterns). Also, other requirement pattern is available for create simple textual requirements. This requirement kind also has been implemented like a requirement pattern. This requirements patterns written in XML files should be based in a certain XML Schema that is included into the tool.

Once the tool has been created we need to test and validate the design so, we have used an internationalization requirements catalog which have a requirements document with few requirements sections and subsections.

Also, we have tested all use case changing input values and check it with expected values or results.

In this document has been made a detailed specification of each component as a content type, a menu option, a vocabulary, etc., to allow to readers know all details for a possible tool improvement or expansion.

To finish, we must say that the main goals of this work have been achieved. Is possible create and maintain a catalog, reuse a complete catalog or requirements of this catalogs, create new requirements and manage requirement's parameters.

Also, the tool has been designed to be improved easily in the future. Creating more custom modules is possible improve PANTALASA-CMS's functionality easily.

Drupal is an excellent CMS in terms of functionality because is very powerful and allows adapt it to your needs but, for this, you need be, nearly, a Drupal expert because you needs type the most of code through Drupal's API.

The main disadvantage that I have found is that Drupal is programmed in PHP and this is very inefficient for complex functionality like copy a set of requirements. Another disadvantage is the amount of database queries needed for get a content because each field of each content type is represented as a database table and you need several queries or a complex query to obtain a content.

For simple functionalities in terms of performance, Drupal is a good option for a tool like PANTALASA-CMS because is very easy install it and, having enough knowledge, not very difficult to develop. Also, Drupal's schema is useful for improve tool step by step and modularly.

3 Introducción y referencias históricas

A la hora de abordar un proyecto de desarrollo de un producto es muy importante tener bien claros y definidos los requisitos del producto. Una mala definición de requisitos podría provocar que las fases posteriores de la metodología de desarrollo usada arrastren esos errores de definición y así una fase tras otra. Conforme el desarrollo avanza es más difícil y costoso corregir esos errores, lo cual provoca que aumente el tiempo de desarrollo y los costes del proyecto al tener que invertir tiempo en solucionar y corregir los errores. De ahí la importancia de tener una buena definición de requisitos desde etapas iniciales del desarrollo.

En una buena definición de requisitos hemos de procurar que los requisitos sean consistentes, correctos y no ambiguos. Aun así esto no asegura el correcto desarrollo del producto y/o la satisfacción del cliente. Además, la especificación de requisitos es una tarea ardua que se puede ver facilitada por la reutilización de especificaciones anteriores. SIREN está concebido para resolver uno de los problemas más importantes que hoy en día afronta la ingeniería del software, la reutilización, para hacer que entre proyectos similares, conjuntos de requisitos puedan ser reutilizados tal cual o con pocas modificaciones.

Por otro lado, a raíz de la creciente tendencia de globalización que está sufriendo la industria en general y la ingeniería del software en particular, se hacen necesarios nuevos métodos que adapten las prácticas y procesos, que siempre han funcionado en un ambiente de trabajo localizado, a la nueva situación. PANGEA propone una extensión del método SIREN, permitiendo la reutilización de requisitos en proyectos de desarrollo global de software. Establece cambios en el modelo de proceso y en la herramienta que da soporte a SIREN. A continuación estudiaremos cuáles son esos cambios, siguiendo los proyectos informáticos de Antonio Muñoz y Jorge Rodríguez [1]y de Alejandro López Lorca [2].

3.1.1 Métodos SIREN y PANGEA

3.1.1.1 Modelo de proceso

El actual modelo de proceso de SIREN [2], que da soporte a la reutilización de requisitos en un entorno localizado de desarrollo de software, no es suficiente cuando deseamos trabajar en un entorno global. Por tanto, PANGEA redefine dicho modelo de proceso añadiendo nuevas actividades y tareas, junto con un nuevo conjunto de roles encargados de desempeñarlas.

En primer lugar podemos encontrar el modelo en espiral del proceso básico de la ingeniería de requisitos (IR [3]) cuyas actividades son cuatro: elicitación, análisis y negociación, documentación y validación, ver Figura 1**¡Error! No se encuentra el origen de la referencia.**:

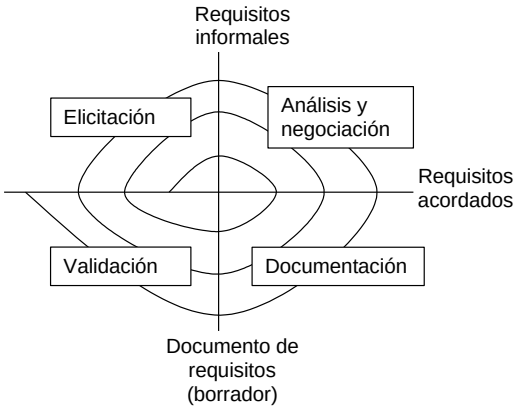


Figura 1 - Modelo en espiral del proceso básico de IR

En SIREN se asume un modelo de proceso en espiral [4] que se enriquece con nuevas actividades para abordar específicamente las cuestiones relacionadas con la reutilización de requisitos. SIREN se concibe como un método de requisitos que puede aportar la consideración explícita de la reutilización a cualquier método de IR. En la Figura 2 se muestra cómo las actividades de SIREN relacionadas con la reutilización extienden el modelo de proceso en espiral de la **¡Error! No se encuentra el origen de la referencia.**.

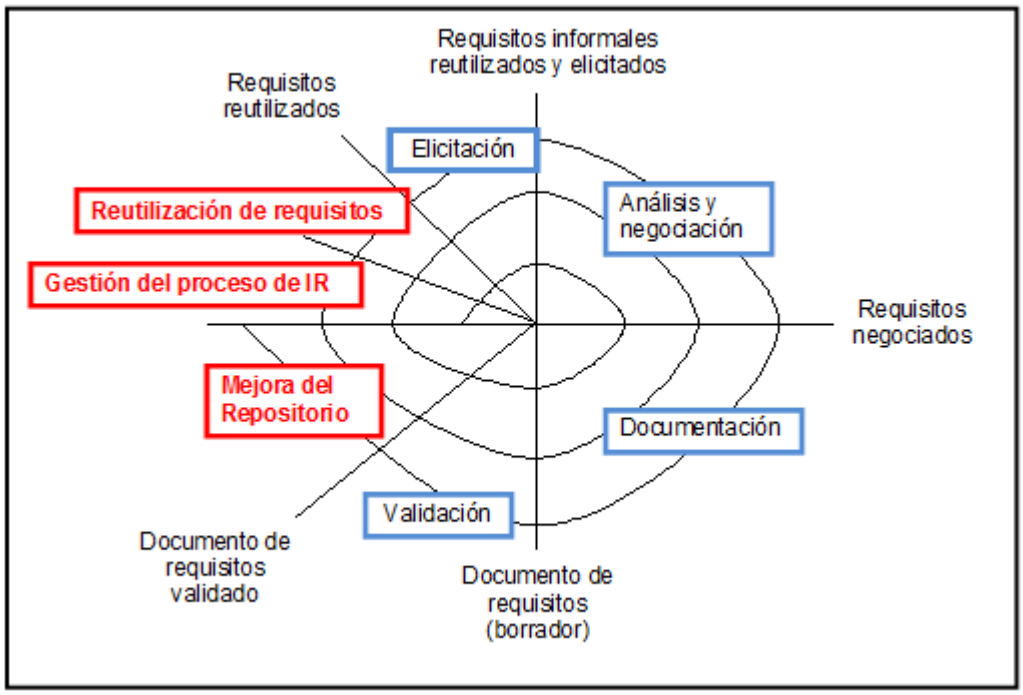


Figura 2 - Modelo de proceso SIREN

Donde:

- **Gestión del proceso de IR**: En el marco de un desarrollo con reutilización, se trata de planificar y monitorizar la ejecución del proceso de requisitos, controlar las desviaciones que se puedan producir durante dicha ejecución y si procede generar lecciones aprendidas que puedan ser de utilidad en futuros desarrollos.
- **Reutilización de requisito**s: Se trata de descubrir y seleccionar los requisitos reutilizables del repositorio que son de aplicación en el proyecto actual.
- **Mejora del repositorio**: Se trata de estudiar posibles mejoras del repositorio de requisitos reutilizables y determinar su inclusión o no en el mismo.

El modelo de proceso que propone PANGEA es un híbrido entre una serie de actividades secuenciales con otras en espiral que proceden básicamente de SIREN, como se muestra en la figura Figura .

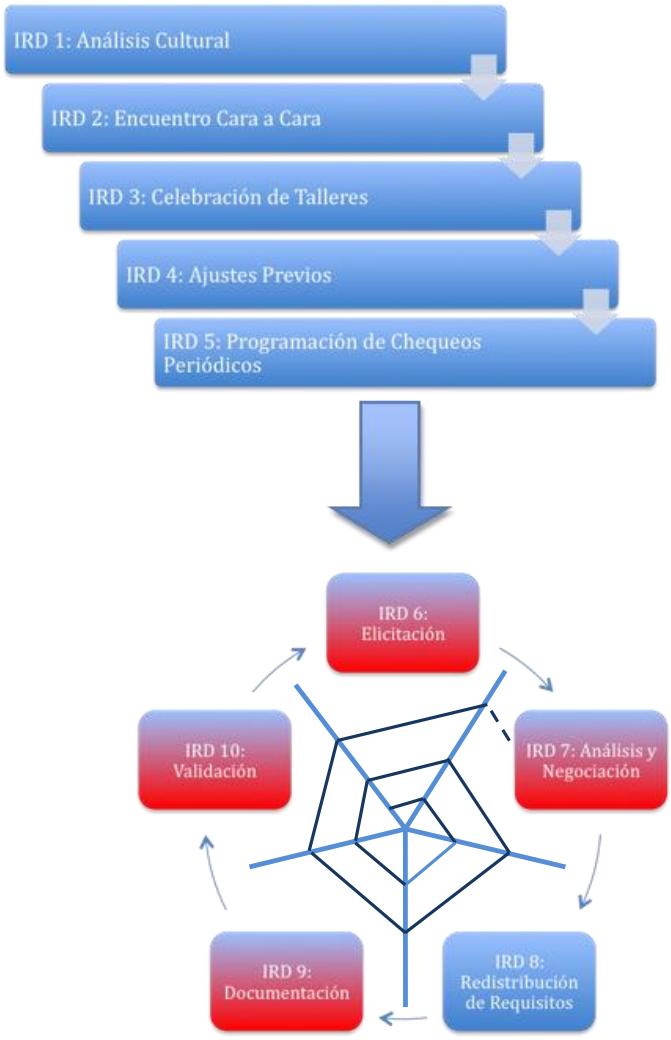


Figura 3 - Modelo de proceso PANGEA

3.1.2 Actividades y tareas

Las actividades de la **¡Error! No se encuentra el origen de la referencia.** marcadas en color rojo son aquellas que permanecen del modelo original de SIREN, mientras que aquellas marcadas en

color azul son totalmente nuevas. Las cinco primeras actividades se realizan de manera secuencial y son de preparación y organización para llevar a cabo con éxito el proceso distribuido, por tanto sólo tendrán que realizarse en la primera iteración. Las cinco últimas conformarían la ejecución efectiva del proceso de ingeniería de requisitos que se realiza en espiral con tantas iteraciones como sean necesarias. A continuación mostraremos una tabla resumen, Tabla 1 y **¡Error! No se encuentra el origen de la referencia.**, explicando cada una de las tareas que intervienen en el modelo de proceso PANGEA, así como las entradas y salidas de cada una de ellas, Tabla 2.

Actividad	Tarea	Descripción
IRD 1:Análisis Cultural	IRD 1.1: Identificación de Nacionalidades IRD 1.2 Recuperación de Informes IRD 1.3 Elaboración de Informes IRD 1.4: Mejora del repositorio de Descripciones Culturales	Análisis de las diferencias culturales entre las culturas de los distintos países que intervienen en el proyecto.
IRD 2: Encuentro Cara a Cara	IRD 2.1 Encuentro cara a cara	Celebración de un encuentro cara a cara entre los participantes del proyecto que les permita establecer las bases para forjar una relación de confianza y aliente la comunicación informal entre ellos.
IRD 3: Celebración de Talleres	IRD 3.1 Celebración de talleres	Puesta en común dentro de cada grupo de trabajo de los informes culturales para conocer los aspectos a tener en cuenta cuando se trate con alguien proveniente de una cultura diferente.
IRD 4: Ajustes Previos	IRD 4.1: Identificación de usuarios clave IRD 4.2: Asignación de Usuarios Clave a equipos de trabajo IRD 4.3: Definición de un idioma oficial IRD 4.4: Compilación de un vocabulario común	Ajustes previos al comienzo del proyecto: identificación de Usuarios Clave, asignación de éstos a los grupos de trabajo, definición de un idioma oficial para el proyecto y construcción de una ontología que sea usada como glosario de términos para el proyecto.
IRD 5: Programación de Chequeos Periódicos	IRD 5.1: Programación de chequeos periódicos	Programación de chequeos periódicos de progresos en los que los grupos deberán informar a los otros sobre la evolución del trabajo.
IRD 6: Elicitación	IRD 6.1: Selección de requisitos reutilizables IRD 6.2: Documentación de requisitos	Reutilización y elicitación de requisitos.
IRD 7: Análisis y Negociación	IRD 7.1: Preparación de la reunión IRD 7.2: Desarrollo de la reunión IRD 7.3: Extracción de conclusiones	Eliminación de inconsistencias del documento de requisitos mediante la discusión síncrona remota de distintos grupos de trabajo.
IRD 8: Redistribución de Requisitos	**IRD 8.1: Propuesta de asignación** **IRD 8.2: Validación de la asignación**	**Redistribución de los requisitos elicitados y reutilizados para su desarrollo y refinamiento a distintos grupos de trabajo.**

Actividad	Tarea	Descripción
IRD 9: Documentación	IRD 9.1: Formalización de documentación	Establecimiento de los documentos de líneas base de requisitos.
IRD 10: Validación	IRD 10.1: Validación de requisitos	Validación con el cliente de los documentos de requisitos.

Tabla 1. PANGEA: Descripción de actividades I

Actividad	Entrada	Salida
IRD 1:Análisis Cultural	Las nacionalidades de los integrantes de los distintos grupos de trabajo participantes en el proyecto.	Informes culturales referentes a todos los países participantes en el proyecto.
IRD 2: Encuentro Cara a Cara	El listado de los participantes en el proyecto.	Se comienza a establecer una relación de confianza entre los participantes remotos en el proyecto.
IRD 3: Celebración de Talleres	Informes culturales sobre los integrantes del proyecto.	Los miembros de los grupos de trabajo conocen las diferencias culturales con otros equipos de trabajo.
IRD 4: Ajustes Previos	Información sobre las localizaciones de los grupos de trabajo, e idiomas dominados por sus integrantes.	Listado de los Usuarios Clave junto con sus localizaciones y los grupos de trabajo que colaborarán con ellos, el idioma oficial del proyecto y el establecimiento de una ontología que modele el dominio de trabajo.
IRD 5: Programación de Chequeos Periódicos	Diferencias horarias entre los grupos de trabajo.	Programación de los informes de progresos de unos equipos de trabajo a otros.
IRD 6: Elicitación	Repositorio de requisitos reutilizables, Usuarios y Usuarios Clave.	Documento de requisitos, provenientes del repositorio y de los usuarios, que puede contener inconsistencias. Orden del día con las cuestiones que hay que discutir sobre requisitos conflictivos.
IRD 7: Análisis y Negociación	Orden del día para la reunión.	Documento de requisitos sin inconsistencias. Actas de la reunión en las que se detallen los aspectos tratados, los registros completos de las conversaciones mantenidas y los resultados de las votaciones llevadas a cabo.
IRD 8: Redistribución de Requisitos	Documento de requisitos sin asignaciones, sólo con indicaciones de qué grupo de trabajo elicitó cada requisito.	Documento de requisitos con asignaciones del tipo grupo de trabajo – conjunto de requisitos.
IRD 9: Documentación	Documento de requisitos sin inconsistencias en desarrollo	Documento de requisitos en forma de línea base

Actividad	Entrada	Salida
IRD 10: Validación	Documento de requisitos pendiente de validación	Documento de requisitos validado y lista de modificaciones a realizar sobre el mismo

Tabla 2. PANGEA: Entradas y salidas de actividades

3.1.2.1.1 Los roles

A continuación explicamos los distintos roles [4]. En primer la Tabla 3 aparecen los roles que define PANGEA junto con una breve descripción a alto nivel de sus responsabilidades.

Rol	Descripción
Responsable del catálogo	Encargado de la coordinación de las actualizaciones de un catálogo dado. Para no sobrecargar de trabajo a una sola persona, en ocasiones este rol es jugado por distintas persona. El grupo de personas que desempeñan este rol constituye el Equipo responsable del catálogo.
Coordinador	Encargado de la coordinación del trabajo de todos los participantes en el proyecto.
Moderador	Modera las reuniones de negociación de requisitos.
Representante de Equipo	Representa a su grupo de trabajo y habla en su nombre con el Coordinador y otros representantes de equipo.
Analista	Desempeña labores de analista propias de cualquier proyecto de desarrollo de software no distribuido.
Usuario Clave	Conoce bien todo el sistema ya aporta el conocimiento necesario para elaborar los documentos de requisitos.
Usuario	Conoce parte del sistema y aporta conocimiento necesario para elaborar los documentos de requisitos.

Tabla 3. PANGEA: Roles

Para cada uno de los roles que define PANGEA vamos a listar cada una de sus responsabilidades, Tabla 4 y Tabla 5. Cabe mencionar que cada participante en el proyecto podrá desempeñar un único rol.

Rol	Responsabilidades
Coordinador	• Estudio de las distintas culturas involucradas en el proyecto y mejora del repositorio de Descripciones Culturales (IRD 1.4). • Organización y dirección de un encuentro cara a cara previo al comienzo del proyecto (IRD 2.1). • Delimitación de la duración de las actividades (en todas las actividades). • Identificación de Usuarios y Usuarios Clave (IRD 4.1). • Asignación de usuarios a grupos de trabajo (IRD 4.2). • Programación de las reuniones diarias de chequeo de progresos (IRD 5.1). • Elaboración de la propuesta de asignación de requisitos a grupos de trabajo (IRD 8.1). • Negociación con los Representantes de Equipos de la propuesta de asignación de requisitos a grupos de trabajo (IRD 8.2). • Publicación de la lista validada de asignación de requisitos a grupos (IRD 8.2). • Congelación de una versión estable de los documentos de requisitos y exportación a un formato que permita su validación (IRD 9.1). • Negociación personal con el cliente de los aspectos a modificar del documento de requisitos (IRD 10.1). • Elaboración y publicación de un documento de cambios a realizar en requisitos por petición expresa del cliente (IRD 10.1).
Representante del Equipo	• Comunicación con otros Representantes de Equipos (durante todas las iteraciones). • Comunicación con el Coordinador (durante todas las iteraciones). • Organización y dirección de talleres culturales dentro de su equipo de trabajo (IRD 3.1). • Creación/mejora de la ontología que modele los conceptos del sistema, dominio o perfil (IRD 4.4). • Informe de progresos a equipos situados en otros husos horarios (durante todas las iteraciones). • Reutilización de requisitos (IRD 6.1). • Elicitación de requisitos (IRD 6.2). • Redacción del documento de requisitos (IRD 6). • Adición de nuevos hilos de discusión al orden del día de las reuniones de negociación (IRD 6). • Estudio del orden del día de las reuniones de negociación (IRD 7.1). • Participación activa en las reuniones de negociación en las que se discuta sobre algún requisito en que ha trabajado algún miembro de su grupo de trabajo (IRD 7.2). • Representación y defensa de las opiniones de los miembros de su grupo de trabajo en las reuniones de negociación (textuales y mediante videoconferencia) (IRD 7.2). • Participación en las votaciones con la opinión acordada dentro del grupo de trabajo (IRD 7.2). • Negociación con el Coordinador de la propuesta de asignación de requisitos a grupos de trabajo (IRD 8.2).

Tabla 4 - PANGEA: Responsabilidades de los roles I

Rol	Responsabilidades
Analista	• Creación/mejora de la ontología que modela los conceptos del sistema, dominio o perfil (IRD 4.4). • Reutilización de requisitos (IRD 6.1). • Elicitación de requisitos (IRD 6.2). • Redacción del documento de requisitos (IRD 6). • Adición de nuevos hilos de discusión al orden del día de las reuniones de negociación (IRD 6). • Estudio del orden del día de las reuniones de negociación (IRD 7.1). • Labores propias de cualquier analista en un proyecto no distribuido (durante todas las iteraciones).
Moderador	• Revisión continua del documento de requisitos en busca de inconsistencias y errores (durante todas las iteraciones). • Preparación del orden del día de las reuniones de negociación (IRD 6). • Convocatoria de reuniones de negociación (IRD 6). • Intervenciones en las reuniones de negociación para mantener el orden (IRD 7.2). • Actualizar los progresos de las reuniones de negociación en la pizarra virtual de la herramienta de negociación (IRD 7.2). • Organización de votaciones (IRD 7.2). • Redacción y publicación de las actas de la reunión (IRD 7.3). • Modificación del documento de requisitos para reflejar los cambios acordados en la reunión (IRD 7.3).
Usuario	• Aporte de conocimiento sobre parte del sistema, dominio o perfil para la elicitación de requisitos (IRD 6.2). • Validación de requisitos (IRD 10.1).
Usuario Clave	• Aporte de conocimiento sobre todo el sistema para la elicitación de requisitos (IRD 6.2). • Validación de requisitos (IRD 10.1).

Tabla 5 - PANGEA: Responsabilidades de los roles II

Dentro de cada grupo de trabajo podemos distinguir dos roles [5], por un lado están los *Analistas* cuyo papel será similar al que desempeñarían en un desarrollo no distribuido, y por otro está el *Representante del Equipo*, que actuaría como cabeza visible dentro del equipo y se comunicaría con otros grupos a través de sus *Representantes de Equipo* y con el *Coordinador*, ver Figura .

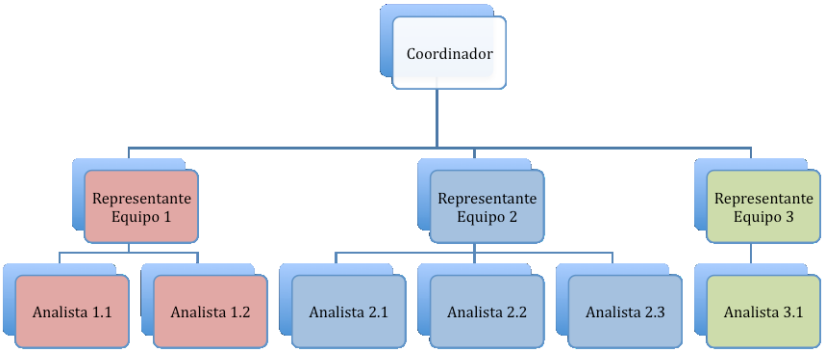


Figura 4 - Jerarquía de los roles en los grupos de trabajo

3.1.2.1.2 Guías de reutilización de requisitos

Una vez establecido el ámbito del proyecto actual debemos buscar en el repositorio si existe algún catálogo de dominio conforme a dicho ámbito [4]. Si es así, posiblemente el proyecto se corresponde con un producto concreto del dominio especificado en el catálogo. En este caso, las búsquedas en el repositorio podrían comenzar buscando en el catálogo del dominio los requisitos con el valor "*alta"* en el atributo *criticidad.* Estos requisitos al ser obligatorios forman parte de cualquier producto de dicho dominio y por tanto deben reutilizarse todos ellos, analizando sus trazas para determinar otros requisitos no obligatorios que también deban formar parte de la especificación del proyecto actual.

Si como resultado de alguna de las búsquedas reutilizamos un requisito que tiene relaciones de traza "*related to"* con otros requisitos, entonces nos encontramos frente a un clúster de requisitos. En este caso debemos considerar la selección de los requisitos implicados en dicho clúster.

La reutilización de requisitos seleccionados implica la resolución de los *puntos de variación* encontrados en tales requisitos:

- Instanciación de los parámetros de los *requisitos parametrizados* con los valores adecuados al proyecto actual.
- Resolución de trazas *exclusive*.
- Resolución de trazas *related to*, que son opcionales.
- Resolución de trazas *required*, que deberían incluirse en condiciones normales.
- Resolución de trazas *padre-hijo*, que deberían incluirse en condiciones normales.

3.1.3 Modelo de referencia de requisitos

3.1.3.1 Conjunto mínimo de atributos

PANGEA extiende el conjunto de atributos de los requisitos de SIREN. Vamos a empezar por los atributos que propone SIREN.

En SIREN, todos los tipos de requisitos tienen asociado un conjunto mínimo de atributos que se muestran a continuación [4]. Están marcados como "Forzosos" aquellos a los que hay que asignar un valor obligatoriamente en el momento de la creación del requisito.

- ***texto*** (Forzoso): la sentencia en lenguaje natural que especifica el requisito.
- ***PUID*** *("Project Unique IDentification") o identificadorÚnico* (Forzoso) del requisito dentro del proyecto.
- ***riesgo***: se debe ponderar cada uno de los requisitos comparándolos con el resto y realizar una estimación del riesgo inherente al requisito (por ejemplo: *alto*, *medio* y *bajo*).
- ***criticidad***: cómo de importante es el requisito para el cliente (por ejemplo *alta* – requisito obligatorio*, media* – requisito recomendable*, y baja* – requisito opcional). En el caso de que exista una relación de exclusividad entre dos o más requisitos obligatorios, se entenderá que sólo uno de ellos debe estar obligatoriamente en la especificación final de requisitos.

- ***prioridad***: el valor de este atributo será establecido por el analista y ayudará a establecer un orden de desarrollo. Los atributos *riesgo* y *criticidad* pueden servir como guía para establecer el valor de la prioridad.
- ***motivación***: indica la razón por la que el requisito está incluido en el proyecto.
- ***estado***: se identifican diez estados en los cuales un requisito se puede encontrar:
  - ***Pendiente de definición***: la redacción del requisito reutilizado o elicitado no se considera definitiva, por ejemplo, se ha reutilizado un requisito parametrizado que no ha sido instanciado o se ha definido un requisito parametrizado al que no se ha asignado un tipo al parámetro. En el caso de que el requisito esté incompleto, este estado también se puede indicar explícitamente en el texto del requisito utilizando el acrónimo TBD (*To Be Determined*).
  - ***Pendiente de revisión****:* el texto del requisito ya se considera completo y está en espera de su revisión (análisis y negociación).
  - ***Pendiente de documentación****:* tras el análisis y la negociación el requisito es aprobado y está pendiente de su documentación, es decir, de su ubicación definitiva en el documento de especificación de requisitos y, opcionalmente, de la revisión del texto y/o atributos del requisito.
  - ***Descartado***: como resultado del análisis y negociación o bien de la validación el requisito es descartado y debe ser renegociado o eliminado.
  - ***Pendiente de validación***: una vez que el requisito ha sido documentado se encuentra en estado pendiente de validación. Después de la validación, es posible que el requisito pase al estado *Descartado* si precisa su renegociación.
  - ***Validado en análisis***: el requisito ha sido aceptado por todos los stakeholders, incluidos los usuarios clave.
  - ***Modelado***: el requisito ha sido modelado en análisis y/o diseño.
  - ***Implementado***: el requisito ha sido codificado.
  - ***Verificado***: una vez implementado se comprueba por parte del equipo de desarrollo que la implementación se ajusta a la especificación del requisito. En caso contrario el requisito requiere la revisión de su implementación
  - ***Validado en implementación***: una vez verificado la implementación del requisito ha sido aceptada por todos los stakeholders, incluidos los usuarios clave. Si la implementación del requisito no es aceptada, el requisito puede requerir su redefinición o la revisión de la implementación. Ver Figura .

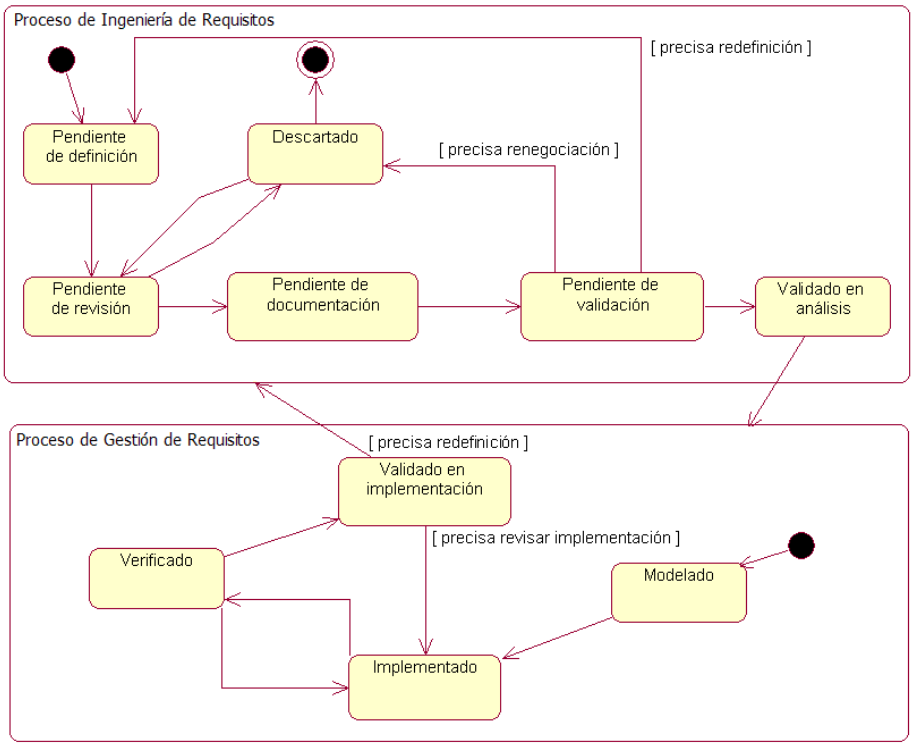


Figura 5 - Diagrama de estados de un requisito

- ***fuente***: (origen del requisito) es un campo de texto en el que se debe poner de dónde procede el requisito. Los requisitos pueden proceder de las necesidades del cliente (y entonces en este atributo se indicará el nombre del usuario clave o típico involucrado), pero habrá otros requisitos que se deriven de restricciones sobre la solución técnica del proyecto, de la legislación vigente o de estándares (y entonces se deberán plasmar las referencias bibliográficas, direcciones web, estándares, etc. correspondientes). Por otra parte este atributo nos servirá para reflejar cuando importemos un requisito del repositorio de qué catálogo proviene y en particular a partir de qué requisito reutilizable.
- ***criteriosValidación****:* indica los criterios de validación que son necesarios para probar el requisito. Normalmente estos criterios se incluyen en el STS (*Software Test Specification*).
- ***propuestoPor****:* indica la persona que solicita que el requisito sea incluido, así en el SyRS hará referencia a un miembro de la organización cliente.
- ***responsable*****:** es aquella persona del equipo de desarrollo responsable de la implementación del requisito.
- ***sección***: indica la sección del documento en la que está especificado el requisito. Según el requisito esté incluido en un documento de SRS, SyRS, STS SyTS o IRS el valor será distinto (por ejemplo, “1.2 alcance”, “3.1.1 Interfaces de usuario”). En este atributo se incluye, además del número de la sección, el nombre de la misma para facilitar búsquedas de requisitos a reutilizar por este atributo.
- ***históricoVersiones****:* autor de la formulación o reformulación del requisito, fecha, versión y texto de la versión enésima.

Partiendo del conjunto anterior de atributos, PANGEA propone nuevos o la modificación de alguno de ellos.

- ***propuestoPor*** (Forzoso): La semántica de este atributo sería la misma que la definida en SIREN pero ahora es de carácter forzoso.
- ***grupoTrabajoFuente*** (Forzoso): Hace referencia al identificador del grupo de trabajo que *elicitó* o reutilizó el requisito.
- ***analistaFuente*** (Forzoso): Hace referencia al nombre del analista que redactó o reutilizó el requisito.
- ***grupoTrabajoDestino***: Hace referencia al identificador del grupo de trabajo al que se ha asignado el requisito para su ampliación o refinamiento.
- ***estado***: Se añadiría un nuevo estado a este atributo:
    - ***PendienteDiscusión***: Nuevo estado que indicará que hay inconsistencias entre grupos de trabajo en relación al requisito y que se debe discutir sobre el asunto en la actividad de negociación de requisitos.
- ***hiloDiscusión***: Hace referencia a los identificadores de las actas de reunión en las que se discutió sobre el requisito.

3.1.3.2 Guías para la definición de requisitos parametrizados

Un requisito se tiene que parametrizar [4] si cuando el cliente adquiere el producto ciertas opciones se pueden elegir entre un conjunto de posibles valores. El parámetro sirve para especificar un *punto de variación* en la especificación de requisitos, esto es, un punto en el que se debe seleccionar entre distintas alternativas para la configuración de un producto concreto. Si todos los tipos posibles deben ser soportados por el producto, y se puede elegir entre uno u otro en tiempo de explotación y no en tiempo de desarrollo, tal característica no se debería poner como un parámetro. Podemos ver un ejemplo:

SRS1. El sistema permitirá la exportación a ficheros externos para la generación de informes.

- *De tipo CVS*
- *De tipo Excel*

El anterior sería un requisito parametrizado si cuando el cliente adquiere el producto se puede elegir entre una u otra clases de exportación (o ambas), quedando el requisito siguiente:

SRS2. El sistema permitirá la exportación a ficheros externos para la generación de informes en [unFormato].

El valor del parámetro “unFormato” podría ser: CVS o Excel

Si no es así, es decir, si las dos exportaciones deben ser soportadas por el producto, no se debería especificar como un requisito parametrizado, sino más bien cómo tres requisitos diferentes con una relación padre-hijo.

SRS1. El sistema permitirá la exportación a ficheros externos para la generación de informes en cualquiera de estos formatos:

- *SRS1.1 Formato CVS*
- *SRS1.2 Formato Excel*

Todo parámetro vendrá caracterizado por tres atributos: *nombre*, *tipo* y *alcance*.

A continuación presentamos un ejemplo de requisito parametrizado extraído del catálogo SEGURIDAD-PDP (*Protección de Datos de carácter Personal*).

SRS2. El responsable del fichero elegirá un [unSoporteFisico] para la realización de copias de seguridad.

Parámetro [nombre="unSoporteFisico", tipo =SOPORTE_FÍSICO {cinta magnética, disco óptico, disco magnético, memoria flash, papel} (1,1), alcance = GLOBAL]

Vemos que el parámetro "unSoporteFisico" se ha definido con alcance global, por tanto, el requisito SRS2 y todos los demás parametrizados con dicho parámetro deben tener el mismo valor cuando se instancien. En este ejemplo, de entre los posibles valores de soporte físico, podemos seleccionar uno ya que no está definido como multivaluado.

Un requisito parametrizado inicialmente se encuentra en estado "Pendiente de definición". Una vez hayan sido instanciados todos los parámetros del requisitos, el estado pasará a "Definido".

3.1.3.3 El metamodelo de trazabilidad

El metamodelo de trazabilidad [2] queda definido de la siguiente manera:

- Trazabilidad **padre/hijo** (*parent/child*): Descripción de un requisito más general por parte de una secuencia de requisitos específicos.
- Trazabilidad **inclusiva** (*inclusive*). Puede ser de dos tipos:
    - **Fuerte** (*required*, A implica B): Dependencia direccional. Para que A se cumpla B se tiene que cumplir necesariamente.
    - **Débil** (*related to*, A relacionado con B): Dependencia bidireccional. Cuando se reutilice A debe considerarse B para reutilización.
- Trazabilidad **exclusiva** (*exclusive*): Relación de exclusividad entre requisitos. Si dos requisitos son exclusivos entre sí, la presencia de uno en la especificación hace que el otro no pueda estar.
- Trazabilidad "**Se materializa en**" (*reifies*): Relación de un requisito y un artefacto del desarrollo, que puede ser una clase, un módulo, un componente, etc.

3.1.3.4 Plantillas de documentos

Por último hablaremos brevemente de las plantillas de documentos a utilizar.

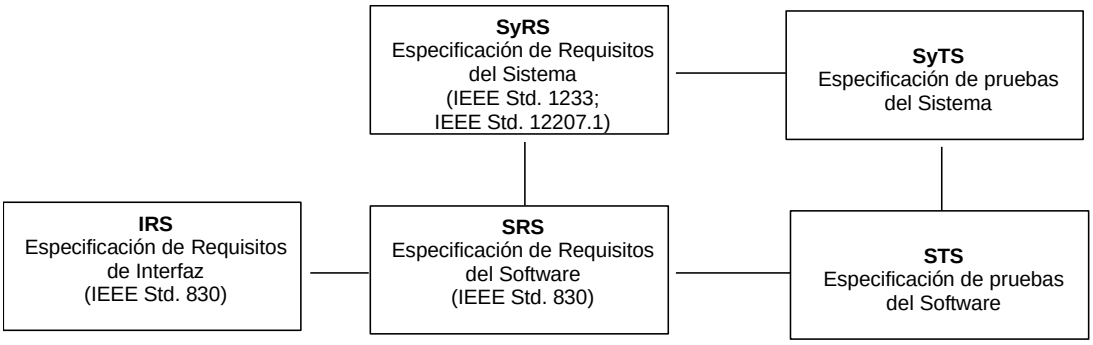


Figura 6 - Jerarquía de plantillas de documentos en SIREN

Aparecen cinco tipos de documentos diferentes que serán utilizados dependiendo de la magnitud y complejidad del proyecto. Los documentos SyRS y SyTS no son necesarios cuando el problema es simple y desde el principio está claro que todos los requisitos serán soportados por el software. El documento SRS es obligatorio y puede dividirse en sub-documentos. El documento IRS es opcional y se utiliza cuando los requisitos de interfaz son muy numerosos con el objeto de evitar que el SRS sea muy extenso. Por último, será necesario un documento STS por cada SRS de la jerarquía.

Esta estructura de documentos es perfectamente compatible para un desarrollo distribuido de software, simplemente habrá que tener en cuenta que el soporte que contenga estas plantillas permita la edición concurrente y distribuida de requisitos, así pues habrá que tender posiblemente a un soporte basado en web.

3.1.4 Otras aproximaciones al soporte de PANGEA

Existe una herramienta de soporte a PANGEA con Wiki Semántica [1] donde hay que destacar la potencia de las búsquedas pero también señalar que son lentas. Además, tiene algunas limitaciones para desarrollar la funcionalidad de PANTALASA.

Con este trabajo fin de grado se quiere explorar si se pueden conseguir soluciones más flexibles sencillas y eficientes haciendo uso de un CMS de libre distribución.

4 Análisis de Objetivos y Metodología

4.1.1 Objetivo del trabajo

El objetivo de este trabajo el desarrollar un prototipo funcional de una herramienta para un repositorio de requisitos basada en la metodología PANGEA mediante un CMS de libre distribución.

Este objetivo general se desglosa en los siguientes objetivos específicos:

1. Desarrollar un prototipo de herramienta CARE de soporte a PANGEA mediante un CMS de libre distribución:
   1.1. Soportar el modelo de referencia de requisitos de SIREN: mecanismos de reutilización de la metainformación asociada a los requisitos (atributos, relaciones de traza); requisitos parametrizados; requisitos exclusivos e inclusivos.
   1.2. Soportar las extensiones que marca PANGEA al modelo de requisitos de SIREN:
      1.2.1.Concurrencia
      1.2.2.Registro de hilos de discusión
      1.2.3.Gestión de usuarios.
   1.3. Soportar un repositorio de catálogos de requisitos SIREN (catálogos y perfiles): creación de catálogos de requisitos reutilizables y reutilización de requisitos procedentes de dichos catálogos en un proyecto concreto, y soportar el acceso distribuido a dicho repositorio: definición distribuida de requisitos, búsquedas.

El CMS seleccionado ha sido Drupal en su versión 7.

4.1.2 Metodología

Para llevar a cabo todo el desarrollo del proyecto se han definido varias tareas a realizar a las cuales se les ha asignado un porcentaje que indica el tiempo y esfuerzo invertido en esa tarea. Podemos verlo en la Tabla 6:

Tarea	Tiempo
Estudio de la metodología PANGEA	10% (30 horas)
Estudio y selección de alternativas	5% (15 horas)
Especificación y diseño de la herramienta con la alternativa seleccionada.	30% (90 horas)
Implementación de un prototipo	35% (105 horas)
Redacción de la memoria	20% (60 horas)

Tabla 6 - Tareas y porcentaje de tiempo

4.1.3 Herramientas y servicios utilizados

En este trabajo fin de grado se ha hecho uso de varias herramientas para facilitar el desarrollo de la herramienta. En la Tabla 7 tenemos un listado con una descripción de cada una de las herramientas utilizadas:

Herramienta	Descripción
Drupal	Distribución base del CMS seleccionado.
Notepad++	Editor de código. Usado para codificar en PHP.

Herramienta	Descripción
Mozilla Firefox	Navegador web para visualizar los resultados.
Google Chroome	Navegador web para visualizar los resultados.
Xampp	Aplicaciones esenciales de desarrollo. Proporciona (entre otras) un servidor web Apache con módulo PHP incluido y una base de datos MySQL para el desarrollo en local de la herramienta.
Microsoft One Note 2013	Herramienta útil para tomar notas.
StarUML	Aplicación para realizar diagramas y modelos.
Assembla	Proveedor de un repositorio SVN para código.
TortoiseSVN	Herramienta para la gestión de los ficheros de la herramienta mediante SVN.
MyEndNoteWeb	Aplicación web para gestionar las referencias de autores y contenido.

Tabla 7 - Herramientas y servicios usados

5 Diseño y resolución

5.1 Especificación del problema

Este trabajo fin de grado consiste en el desarrollo del prototipo de una herramienta de soporte a PANGEA basado en CMS de libre distribución.

Los aspectos que debemos tener en cuenta para el desarrollo de la herramienta son, principalmente, la gestión de catálogos de requisitos y su reutilización. Además, se debe poder ampliar la herramienta en un futuro y se quiere que se haga de forma sencilla.

El método de PANGEA define varios roles de usuarios donde cada uno realizas ciertas tareas por lo que deberemos realizar un control de roles y permisos. Normalmente, un CMS tiene esta característica implementada lo que facilitaría mucho el desarrollo. No obstante, en control de acceso y gestión de permisos que requiere esta herramienta no es trivial y será uno de los aspectos a tener en cuenta a la hora de seleccionar un CMS u otro.

Debemos tener en cuenta que la herramienta será usada en un entorno distribuido por lo que es necesario realizar un control de concurrencia. Normalmente, los CMS son sitios webs y es el propio servidor quien gestiona la concurrencia de peticiones y, a más alto nivel, se encarga el CMS.

En cuanto a persistencia tenemos que decir que los catálogos, requisitos, documentos y demás objetos de negocio han de tener persistencia en el sistema. Un CMS, como su propio nombre indica (Content Management System), es un gesto de contenidos por lo que la gestión de la persistencia es llevada a cabo por el propio CMS.

Otro tipo de cuestiones más relacionadas con el método PANGEA, como pueden ser la gestión de parámetros y la instanciación en requisitos, hilos de discusión de requisitos, grupos de trabajo en proyecto, patrones de requisitos, etc., han de tenerse encuenta debido a que este tipo de funcionalidad es algo más concreto y puede que no sea sencillo encotrar alguna extensión del CMS que lo permita por lo que habrá que contemplar la posibilidad de realizar la implementación de dichas funcionalidades. Habrá, por tanto, que tener en cuenta estas características a la hora de la elección.

5.2 Análisis y selección de alternativas

Para llevar a cabo el diseño de nuestra herramienta se parte de la restricción de diseño de usar un CMS de libre distribución.

Un CMS o sistema de gestión de contenidos (del inglés, *Content Management System*), es una aplicación para la creación y gestión de contenido, principalmente en la web. La funcionalidad de un CMS abarca desde la creación de sitios web para gestionar catálogos de productos, documentos, etc, hasta la creación foros, blogs y redes sociales.

En internet podemos encontrar gran cantidad de sitios web implementados con CMS, como por ejemplo:

http://www.ubuntu.com/, http://appdeveloper.intel.com/en-us/, http://www.whitehouse.gov/, https://mozillalabs.com.

El uso de un CMS se hace a través de una interfaz en el navegador web que controla bases de datos aunque también hay CMS que proporcionan un API para acceder a su funcionalidad desde sitios y aplicaciones externas. Generalmente existen y pueden definirse una serie de roles con distintos permisos para la creación y gestión de contenido.

Actualmente existen gran variedad de CMS pero muchos de ellos han quedado abandonados o desactualizados. La lista de CMS disponibles es muy extensa y sería una labor tediosa hacer una comparativa entre todos ellos. En internet podemos encontrar cantidad de sitios web con listas de los CMS más usado del mercado [6] [7] [8] [7] [9].

Por tanto, en lugar de ello vamos a realizar una comparativa entre los CMS de libre distribución que cuentan con mayor apoyo y reconocimiento de la comunidad de desarrolladores basándonos en los resultados obtenidos al consultar las listas de los CMS más usados [7] [6-8] [9]

Si navegamos por internet en distintos sitios web usados y mantenidos por la comunidad de desarrolladores [9] [10] podemos ver información acerca de cuáles son los CMS más usados actualmente. La mayoría de estos sitios web establecen como orden de uso a Wordpress [11], Joomla [12] y Drupal {Drupal, , Drupal - Open Source CMS}. Otro CMS open source que está adquiriendo importancia en los últimos años es Magnolia CMS [13], un CMS Java muy potente y profesional.

5.2.1 Requisitos de la herramienta PANTALASA-CMS

Antes de pasar a comparar estos tres CMS vamos a definir los requisitos básicos obligatorios que nuestra herramienta ha de satisfacer. Esos requisitos son:

- Definir roles de usuarios
- Control de acceso a usuarios según roles
- Control de acceso a usuarios según contenido
- Gestionar contenidos: crear nuevos contenidos, modificarlos, eliminarlos y realizar consultas sobre ellos.
- Interacción social
- Libre distribución
- Que sea fácilmente modificable y se pueda ampliar su funcionalidad en un futuro.

5.2.2 Wordpress

Wordpress es un CMS orientado a la creación de blogs con licencia GPL y desarrollado en PHP y MySQL. Su principal funcionalidad es la de la creación de blogs o bitácoras con entradas ordenadas por fecha [14].

Tiene actualizaciones automáticas debido a la gran comunidad de desarrolladores que hay detrás de este proyecto [15].

Permite la administración y gestión de distintos usuarios donde cada uno de los usuarios tiene un determinado rol y perfil personalizable.

WordPress está basado en instalaciones estándar de PHP + MySQL, disponibles en todos los servidores. Es famosa su instalación en 3 pasos. Las actualizaciones se realizan desde el mismo panel de administración "a un clic". Además facilita la importación y exportación de los sitios mediante la utilidad propia al efecto [15].

Dispone de módulos externos que se pueden incluir para aumentar la funcionalidad. Actualmente es el CMS con más módulos disponibles, cuenta con más de 14000.

Las páginas del sitio se pueden generar de forma dinámica lo que reduce el tiempo de desarrollo de forma considerable.

WordPress ha evolucionado enormemente y actualmente es el CMS utilizado por los sitios de más tráfico de la red como Techcrunch y otros. Recientes estudios han desestimado leyendas urbanas acerca de los modernos CMS y demostrado que lo que marca la diferencia es la optimización y la capacidad del servidor [15].

Soporta más de 50 idiomas y permite interconectar los blogs y bitácoras. Además soporta RDF.

Es un CMS muy fácil de aprender, quizás, el más rápido orientado a blogs y bitácoras.

5.2.3 Joomla

Joomla es un CMS de código abierto y está escrito en PHP, usa bases de datos MySQL y se distribuye bajo la licencia GPL. En palabras menos técnicas, es un software libre, que no paga licenciamiento y se basa en herramientas similares, que no generan costos de licencias [16].

Consta de plugins y módulos aunque son más escasos que los que tienen Drupal o Wordpress pese a la gran cantidad de tiempo que lleva en el mercado y la gran comunidad de usuarios y desarrolladores con la que cuenta.

Su gestión de usuarios y roles distintos es muy limitada e ineficiente por lo que no es de gran utilidad para sitios web con mucha carga de usuarios y permisos.

Este CMS estaría orientado a proyectos simples y orientado a la publicación de contenidos. "*Si vuestra web es una web estándar y básica que no va a crecer mucho en número de prestaciones ni secciones os recomiendo Joomla [17]".*

Tal vez la seguridad sea uno de los principales caballos de batalla de los detractores del Joomla. En realidad, la seguridad dependerá del administrador de la página. Simplemente debe estar muy atento a las actualizaciones y parches que salgan (normalmente cada dos o tres meses). Pero si hay descuido por parte del Webmáster y nunca se actualiza, las vulnerabilidades estarán presentes y la puerta abierta para que la página sea atacada [16].

5.2.4 Drupal

Drupal es con diferencia el CMS más complicado de usar a costa de ampliar considerablemente las posibilidades de configuración y personalización que ofrece. Por ejemplo, a la hora de configurar el acceso a una URL de una opción de navegación o una funcionalidad es necesario realizar la implementación de varias funciones: una para que la opción de navegación sea accesible a través de una URL, otra función que realice la implementación de esa opción de navegación y otra función que realice el control de acceso a dicha funcionalidad. Además,

todas estas funciones pueden incluir parámetros de entrada por lo que habría que realizar un control sobre qué parámetros se usan en cada función. Aun se puede complicar más al hacer uso de la API Form de Drupal para crear formularios ya que habrá que usar la función que proporciona Drupal y además, implementar nuestras propias funciones.

Como vemos es muy complicado y laborioso realizar este tipo de funcionalidades pero ofrece potencia y versatilidad.

Dispone de una gran cantidad de módulos para todo tipo de funcionalidad aunque con los que trae en el core son suficientes para realizar casi cualquier cosa.

El sistema de módulos, aplicaciones, plugins, componentes, add-ons, etc… (usamos distinto vocabulario para las distintas plataformas) es diferente.

Los módulos en Drupal tienen más granularidad, es decir si se quiere montar, por ejemplo, un catálogo de productos con Drupal tienes distintos módulos para realizarlo, uno que te sube las imágenes, uno que te permite crear distintos campos, uno que te permite mostrar distintas vistas del producto, etc… es decir necesitamos varios módulos para construir un catálogo, hay más trabajo pero el resultado es muy manejable [18].

En Wordpress o Joomla es distinto con un solo módulo tienes un catálogo, que puedes configurar y modificar un poco pero es más difícil de adaptar a casos de necesidades específicas.

El código es bastante limpio y eficiente y consta de una buena separación con las vistas.

Está principalmente orientado a grandes proyectos de comunidades de usuarios y en este aspecto resulta un CMS muy potente.

Permite, por tanto, definir varios tipos de usuarios y establecer permisos acorde a las necesidades del proyecto. Drupal no solo permite realizar un sitio web a medida sino un tablero de administración a medida [18].

Su seguridad es más alta que en los otros CMS y permite realizar una gestión muy avanzada de control de acceso y chequeo de permisos.

También tiene soporte a gran cantidad de idiomas y soporta varios gestores de bases de datos.

Soporte de RDF y ontologías y extensible a web semántica mediante módulos.

Es un CMS de carácter general muy personalizable y extensible para cualquier tipo de proyecto.

5.2.5 Magnolia-CMS

Este CMS no es muy conocido actualmente por los desarrolladores debido a que se requiere mayor conocimiento de programación al ser un CMS hecho en JAVA. Dispone de versiones libres y versiones comerciales.

Su instalación y configuración es mucho más complicada que con los CMS vistos anteriormente y se requiere de un servidor más potente para su instalación.

Pese a ser extensible mediante módulos, el número de estos es mucho menor que el de los otros CMS y aunque permite desarrollar módulos, es mucho más complicado hacerlo que en Wordpress o Drupal.

La creación de tipos de contenido es mucho más compleja que en otros CMS ya que no tenemos representación visual de los resultados hasta que no publicamos el contenido. De igual modo ocurre con la creación de formularios de edición también denominados "diálogos".

Una de las ventajas que presenta es que todo el contenido se implementa como un componente y esto permite reutilizarlo de forma sencilla indicando una simple referencia al componente genérico.

Las características que presenta este CMS no mejora las del resto de CMS vistos anteriormente para la implementación de nuestra herramienta pero merece la pena citar brevemente este CMS por usar una tecnología diferente y más avanzada para su implementación como es el lenguaje JAVA.

5.2.6 Tabla comparativa

Podemos ver en la Tabla 8 un resumen de las características de cada CMS:

	Wordpress	Joomla	Drupal	Magnolia-CMS (CE)
Versión	3	3	7	5
Gestión de Usuarios	SI (limitado)	SI (limitado)	SI	SI
Extensible	SI	SI	SI	SI
Orientación (funcionalidad)	Simple	Simple	Cualquiera	Cualquiera
Cantidad de módulos/plugins	+14000	+7000	+8000	(Muy pocos)
Soporte Ontologías	SI	-	SI	-
Facilidad de uso	Fácil	Muy fácil	Difícil	Muy difícil
Programación	PHP	PHP	PHP	JAVA
Bases de datos	MySQL	MySQL, SQLServer2008 y SQLAzure	SQLite, MySQL/MariaDB y PostgreSQL, SQL Server y Oracle	MySQL, PostgreSQL, Oracle or MS SQL Server
Sistema Operativo	Todos	Todos	Todos	Todos
Multi-idiomas	SI	SI	SI	SI

Tabla 8 - Tabla comparativa de CMS

5.2.7 Conclusiones

Como vemos, Joomla no tiene soporte conocido para ontologías por lo que quedaría descartado de nuestro proyecto pese a su facilidad de uso y la cantidad de opciones que nos permite.

La siguiente opción más deseable sería Wordpress por la cantidad de módulos que posee y su facilidad de uso. No obstante hemos mencionado las limitaciones que presenta al ser un CMS orientado a bitácoras, foros y sitios con funcionalidad relativamente simple, lo que limita el ámbito de nuestra herramienta y su implantación en empresas de desarrollo de software.

Además, ese carácter de creación de blog hace que, aunque puede ser totalmente ampliable mediante módulos, su rendimiento se ve afectado al no ser esos módulos orientados a blogs o bitácoras.

Drupal tiene un carácter general y presenta un framework de desarrollo con opciones de ampliación ilimitadas además de ser un CMS puro.

Está destinado a comunidades de usuarios y consta con un core que ofrece la mayoría de funcionalidades necesarias para un sitio web o aplicación de cualquier tipo.

La gestión de usuarios y permisos es muy rica pudiéndose mejorar aún más mediante módulos para poder adaptarlo a todas las necesidades. La comprobación de acceso y permisos para las funciones del módulo también es muy elegante y sencilla de implementar. La edición del código, pese a ser compleja, es muy potente y clara ya que permite modularizar y dividir todo el código de forma que quede muy organizado y claro

En el ámbito de nuestra herramienta necesitamos un CMS de carácter general soportado por una gran comunidad de usuarios que ofrezca una gran potencia en cuanto a seguridad y rendimiento y sea perfectamente ampliable, es decir, que no sólo se pueda ampliar y realizar actualizaciones sobre la distribución de Drupal que sea posible ampliar nuestra propia herramienta mediante otros módulos.

Por todos estos motivos nos decantamos como opción de desarrollo por Drupal en su versión 7.

5.3 Diseño

5.3.1 Modelo de Casos de Uso

5.3.1.1 Actores

Responsable del catálogo: Este es un nuevo rol creado para realizar tareas administrativas dentro del repositorio. Además puede realizar todas las tareas disponibles por lo que se le otorga poder total.

Coordinador: Este actor tiene tareas administrativas dentro de un proyecto o varios y se encarga de gestionar documentos de requisitos.

Moderador: Este actor se encarga la gestión de requisitos de un proyecto al igual que lo hace un Representante de Equipo o un Analista. La responsabilidad adicional que tiene este actor es la de gestionar las distintas versiones de los documentos de requisitos. También tiene la responsabilidad de crear los hilos de discusión para los requisitos donde se debatirán y tratarán aspectos de un requisito.

Representante de Equipo y Analista: Estos dos actores tienen tiene la responsabilidad de gestionar proyectos y todos sus requisitos y documentos. Pueden consultar los catálogos y aplicarlos a sus proyectos. En nuestro proyecto no existirán diferencias entre uno y otro debido a la funcionalidad que se exige a la herramienta, sin embargo, definiremos estos dos roles de forma separada para facilitar la ampliación futura de la herramienta.

Usuario y Usuario clave: Estos dos actores no tendrán funcionalidad en el sistema. Aparecen en la metodología de PANGEA como actores que proporcionan información y conocimiento sobre el sistema y además pueden validar los requisitos. Pueden aparecer referenciados en los requisitos como stakeholder que propone el requisito. Además pueden consultar información del proyecto, documentos y sus requisitos.

Objetivos de Usuario

- Responsable del catálogo
  - Gestionar catálogos
  - Gestionar tipos de parámetros
- Coordinador
  - Crear versión base/estable de los documentos de requisitos
  - Exportar documento de requisitos a otros formatos
  - Creación de proyectos
  - Gestión de grupos de trabajo
  - Gestión de Usuarios y Usuarios Clave
- Moderador
  - Gestionar documentos propios
  - Gestionar hilo de discusión
  - Supervisar y administrar los mensajes de hilos.
- Representante de Equipo/Analista

  - Redactar un documento de requisitos
  - Consulta de documento de requisitos del proyecto
  - Crear una ontología de un dominio o concepto
  - Participar en hilos de discusión
  - Reusar un catálogo de requisitos
  - Consultar los catálogos de requisitos
- Usuario / Usuario Clave
  - Consulta de información de proyecto.

A continuación se muestran los diagramas de casos de uso:

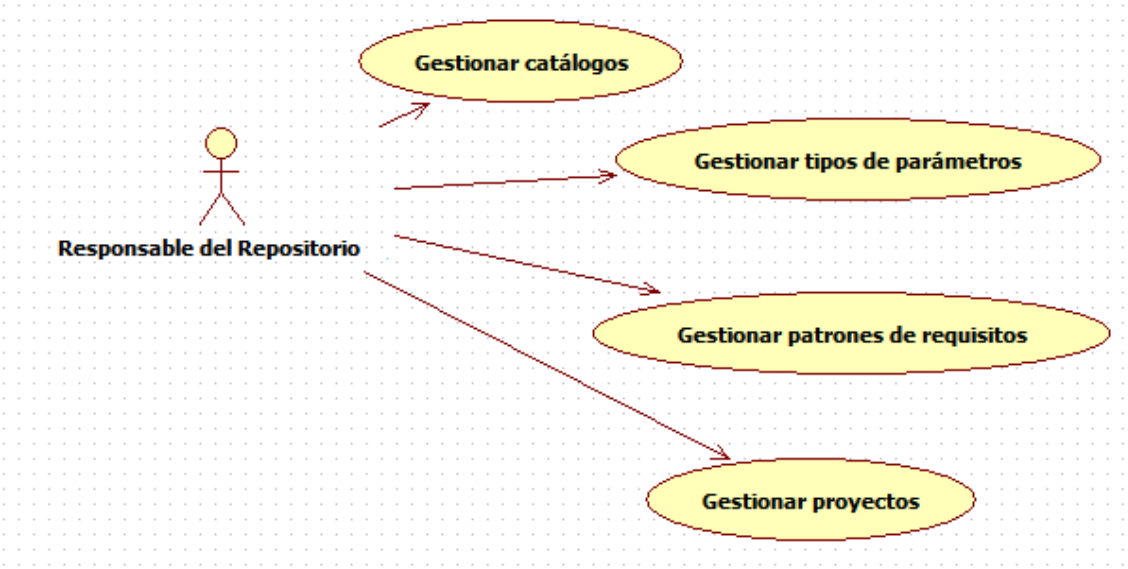


Figura 7 - Casos de uso del responsable del repositorio

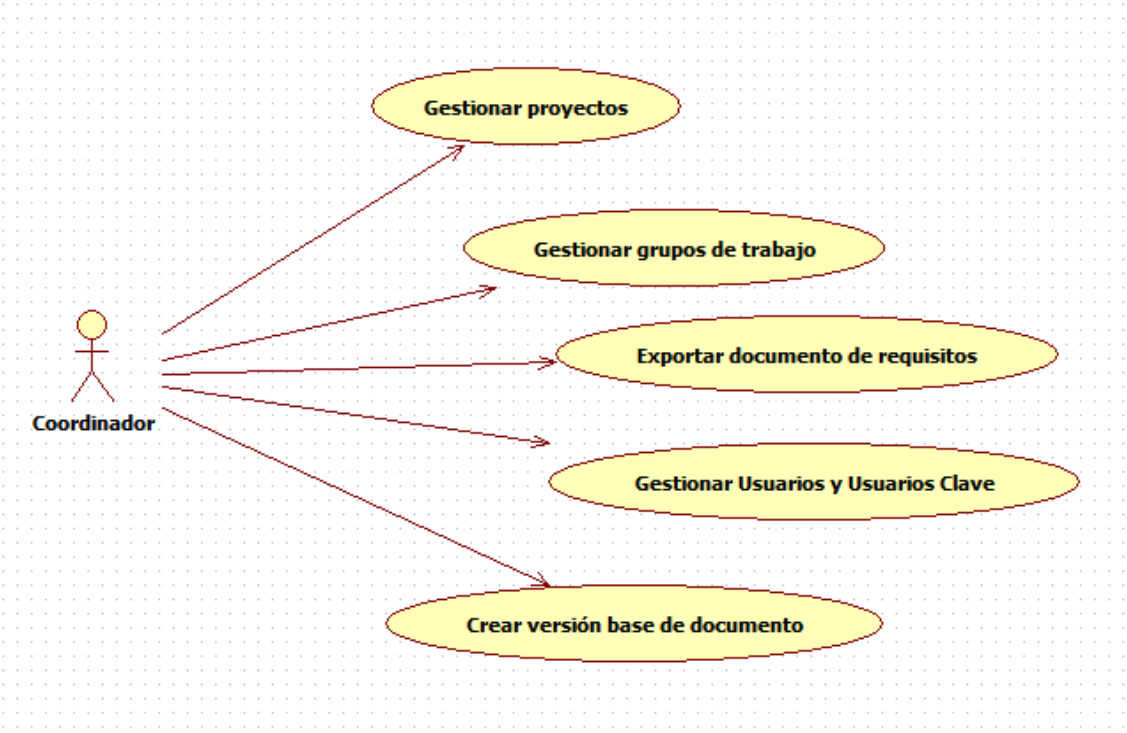


Figura 8 - Casos de uso del coordinador

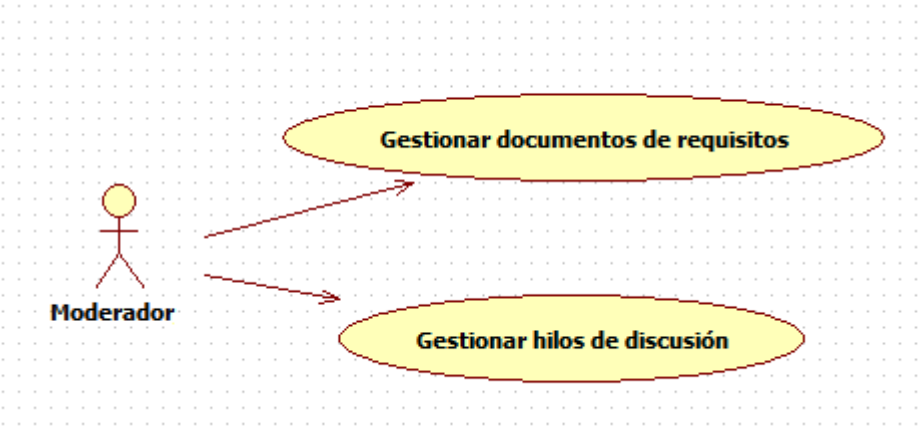


Figura 9 - Casos de uso del moderador

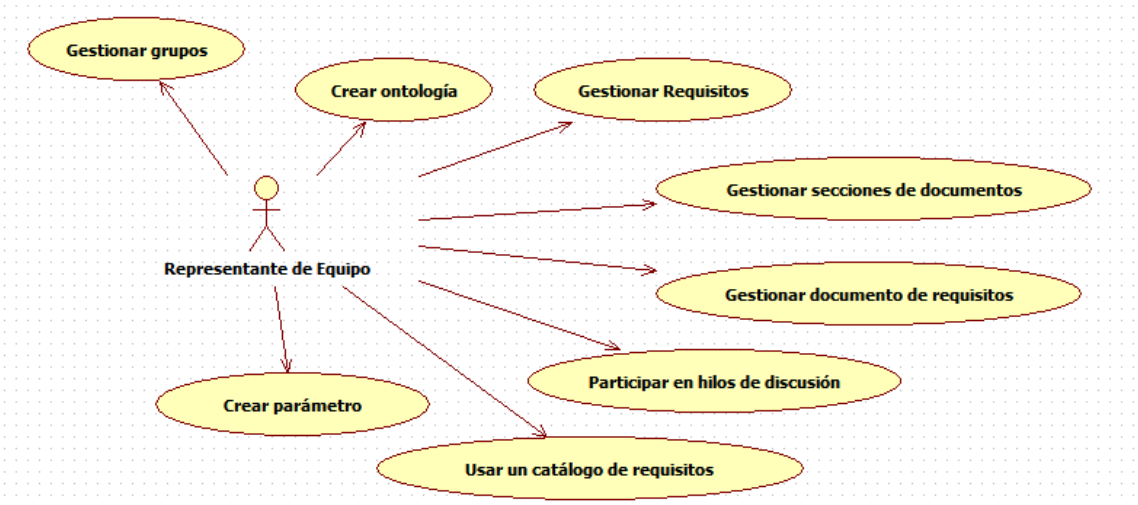


Figura 10 - Casos de uso del representante de equipo

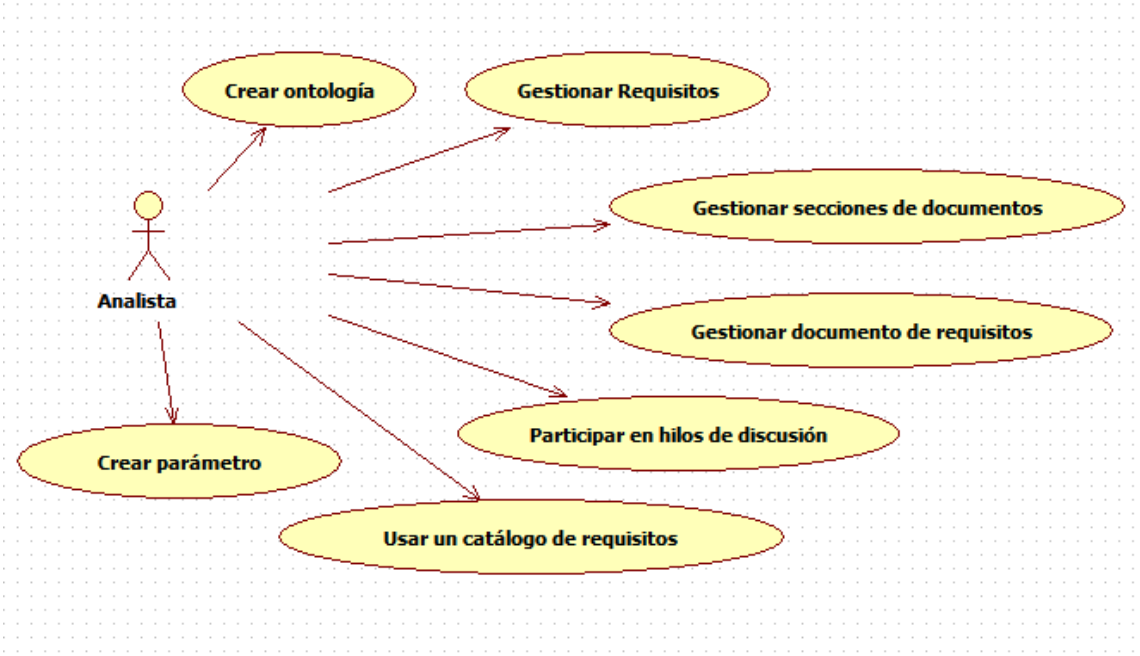


Figura 11 - Casos de uso del analista

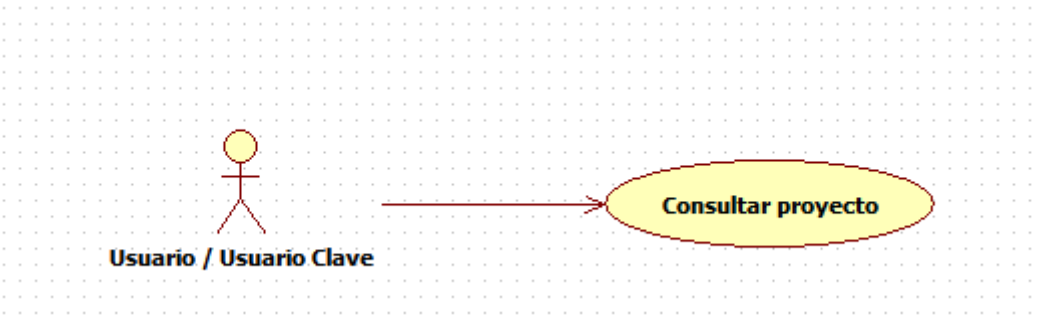


Figura 12 - Casos de uso del usuario y usuario clave

5.3.1.2 Especificación de los Casos de Uso

A continuación se van a realizar la especificación completa de los casos de uso:

5.3.1.2.1 CDU 1 – Gestionar catálogos

Caso de Uso	**CDU 1 – Gestionar catálogos**
Descripción	Operaciones de creación, consulta, modificación y borrado de catálogos de requisitos.
Actores	Responsable del catálogo
Asunciones	• El Responsable del catálogo está autenticado en el sistema.
Pasos	1. El Responsable del catálogo accede al repositorio 2. El Responsable del catálogo pulsa sobre la opción de Catálogos 3. El Sistema muestra un listado con los catálogos
Variaciones	3.a.1 El Responsable del catálogo pulsa sobre la opción de "Create new catalog" 3.a.2 El Responsable del catálogo introduce el nombre y la descripción del catálogo y pulsa "Submit". 3.a.3 El Sistema registra el nuevo catálogo y añade la fecha y hora de creación y el usuario que lo creó. 3.a.4 Fin del caso de uso 3.b.1 El Responsable del Catálogo pulsa sobre la operación "Edit". 3.b.2 El sistema muestra un formulario con el nombre y la descripción del catálogo. 3.b.3 El Responsable modifica los datos que quiera y pulsa sobre "Submit". 3.b.4 El Sistema guarda los cambios. 3.b.5 Fin del caso de uso 3.c.1 El Responsable del Catálogo pulsa sobre la operación "Delete". 3.c.2 El Sistema elimina todos los requisitos del catálogo y todos los documentos. 3.c.3 Fin del caso de uso
Req. No-funcionales	-
Cuestiones	-

Tabla 9 - CDU 1 - Gestionar catálogos

5.3.1.2.2 CDU 2 – Gestionar tipos de parámetros

Caso de Uso	CDU 2 - Gestionar tipos de parámetros
Descripción	Operaciones de consulta, creación, edición y borrado sobre tipos de parámetros.
Actores	Responsable del catálogo
Asunciones	• El Responsable del catálogo está autenticado en el sistema.
Pasos	1. El Responsable del Catálogo accede al repositorio 2. El Responsable del Catálogo pulsa sobre la opción “Param types” 3. El Sistema muestra un listado con los tipos de parámetros existentes.
Variaciones	3.a.1 El Responsable del Catálogo pulsa sobre “Create new param type” 3.a.2 El Responsable del Catálogo introduce el nombre, la descripción del nuevo tipo y el campo clave y pulsa “Submit”. 3.a.3 El Sistema crea el nuevo tipo. 3.a.4 Fin del caso de uso 3.b.1 El Responsable del Catálogo pulsa sobre un tipo de parámetro. 3.b.2 El Sistema muestra las operaciones disponibles sobre un tipo de parámetro para el Responsable del Catálogo. 3.b.3 El Responsable del Catálogo pulsa sobre la opción de “Edit” 3.b.4 El Sistema muestra el nombre, la descripción del tipo de parámetro y el campo clave en un formulario de edición. 3.b.5 El Responsable del Catálogo modifica los datos que quiera y pulsa sobre “Submit”. 3.b.6 El Sistema modifica los datos del tipo de parámetro. 3.b.7 Fin del caso de uso 3.b.3.a.1 El Responsable del Catálogo pulsa sobre la opción de “Delete”. 3.b.3.a.2 El Sistema elimina el tipo. 3.b.3.a.3 Fin del caso de uso
Req. No-funcionales	-
Cuestiones	Los tipos de datos usados no se borrarán aunque ya no se usen.

Tabla 10 - CDU 2 - Gestionar tipos de parámetros

5.3.1.2.3 CDU 3 – Gestionar proyectos

Caso de Uso	**CDU 3 – Gestionar proyectos**
Descripción	Operaciones de consulta, creación, edición y borrado sobre proyectos.
Actores	Responsable del repositorio
Otros actores	Coordinador
Asunciones	• El Responsable del repositorio está autenticado en el sistema.
Pasos	1. El Responsable del repositorio accede al repositorio 2. El Responsable del repositorio pulsa sobre la opción de "Projects" 3. El Sistema muestra un listado de Proyectos sobre los que el usuario tiene acceso. 4. El Responsable del repositorio pulsa sobre la opción de "Create new project". 5. El Responsable del repositorio introduce el nombre y la descripción del nuevo proyecto 6. El Sistema registra la fecha y hora de creación y el usuario que lo ha creado. 7. El Sistema crea el nuevo proyecto.
Variaciones	4.a.1 Fin del caso de uso. 4.b.1 El Coordinador pulsa sobre un proyecto. 4.b.2 El Sistema muestra las opciones disponibles sobre un proyecto para un Coordinador. 4.b.3 El Coordinador pulsa sobre la opción "Edit". 4.b.4 El Sistema muestra en un formulario de edición el nombre y la descripción del proyecto. 4.b.5 El Coordinador modifica los datos quiera y pulsa sobre "Submit". 4.b.6 El Sistema guarda los cambios. 4.b.7 Fin del caso de uso 4.b.3.a.1 El Coordinador pulsa sobre la opción "Delete" 4.b.3.a.2 El Sistema elimina todos los requisitos y documentos del proyecto y eliminará el proyecto. 4.b.3.a.3 Fin del caso de uso
Req. No-funcionales	-
Cuestiones	-

Tabla 11 - CDU 3 - Gestionar proyectos

5.3.1.2.4 CDU 4 - Gestionar grupos de trabajo

Caso de Uso	CDU 4 – Gestionar grupos de trabajo
Descripción	Creación de grupos de trabajo, edición y consulta de los mismos.
Actores	Coordinador
Asunciones	• El Coordinador está autenticado en el sistema.
Pasos	1. El Coordinador accede al repositorio. 2. El Coordinador pulsa sobre la opción de “Groups” 3. El Sistema muestra las operacion de “Create group”. 4. El Coordinador pulsa sobre la opción de “Create group” 5. El Sistema muestra un formulario de creación. 6. El Coordinador introduce el nombre y la descripción del grupo. 7. El Coordinador añade usuarios al grupo. 8. El Coordinador pulsa sobre “Submit” 9. El Sistema crea el grupo de trabajo.
Variaciones	4.a.1 El Coordinador pulsa sobre la opción “Groups” 4.a.2 El Sistema muestra un listado de grupos de trabajo. 4.a.3 El Coordinador selecciona un grupo de trabajo. 4.a.4 El Sistema muestra las operaciones disponibles sobre el grupo para un Coordinador. 4.a.5 El Coordinador pulsa sobre la opción “Edit” 4.a.6 El Sistema muestra un formulario de edición con el nombre, descripción, usuarios. 4.a.7 El Coordinador modifica los datos que quiera y pulsa sobre “Submit” 4.a.8 El Sistema guarda los cambios. 4.a.9 Fin del caso de uso
Req. No-funcionales	-
Cuestiones	-

Tabla 12 - CDU 4 - Gestionar grupos de trabajo

5.3.1.2.5 CDU 5 - Exportar documento de requisitos

Caso de Uso	CDU 5 – Exportar documento de requisitos
Descripción	Exportar documentos de requisitos a formato XML.
Actores	Responsable del repositorio
Otros actores	Coordinador, moderador, analista, representante de equipo
Asunciones	• El Responsable del repositorio está autenticado en el sistema.
Pasos	1. El Responsable del repositorio selecciona el documento a exportar 2. El Responsable del repositorio pulsa sobre la opción “export”. 3. El sistema genera un fichero XML y permite su descarga.
Variaciones	-
Req. No-funcionales	-
Cuestiones	-

Tabla 13 - CDU5 - Exportar documento de requisitos

5.3.1.2.6 CDU 6 – Gestionar usuarios de un proyecto

Caso de Uso	**CDU 6 – Gestionar usuarios de un proyecto**
Descripción	Creación, modificación y borrado de usuarios clave.
Actores	Coordinador
Asunciones	• El Coordinador está autenticado en el sistema.
Pasos	1. El Coordinador accede al repositorio. 2. El Coordinador pulsa sobre la opción de “Users” 3. El Sistema muestra las operaciones de “Create user”. 4. El Coordinador pulsa sobre la opción de “Create user”. 5. El Sistema proporciona un formulario de creación de usuario donde pide el nombre del usuario y el tipo de usuario. 6. El Coordinador introduce el nombre del usuario el tipo de usuario y pulsa “Submit”. 7. El Sistema crea el nuevo usuario.
Variaciones	4.a.1 El Coordinador pulsa sobre la opción de “Users”. 4.a.2 El Sistema muestra un listado de usuarios junto con operaciones disponibles en cada uno. 4.a.3 El Coordinador pulsa sobre la opción “Edit”. 4.a.4 El Sistema proporciona un formulario de edición con los datos del usuario. 4.a.5 El Coordinador modifica los datos que quiera y pulsa sobre “Submit”. 4.a.6 El Sistema guarda los cambios. 4.a.7 Fin del caso de uso
Req. No-funcionales	-
Cuestiones	-

Tabla 14 - CDU 6 - Gestionar usuarios de un proyecto

5.3.1.2.7 CDU 7– Crear versión base de documento

Caso de Uso	CDU 7 – Crear versión base de documento
Descripción	Cuando el Coordinador crea que un documento de requisitos puede servir como punto de partida para el desarrollo se crea una versión base no modificable.
Actores	Coordinador
Asunciones	• El Coordinador está autenticado en el sistema. • Existe un documento d
Pasos	1. El Coordinador pulsa sobre la opción de “Projects” 2. El Sistema muestra un listado de Proyectos sobre los que el usuario tiene acceso. 3. El Coordinador pulsa sobre un proyecto. 4. El Sistema muestra un listado de documentos del proyecto. 5. El Coordinador selecciona un documento d de la lista de documentos disponibles. 6. El Coordinador selecciona la opción de crear documento base a partir de d. 7. El Sistema guarda el usuario que hizo la versión (Coordinador), la fecha y el número de versión.
Variaciones	-
Req. No-funcionales	-
Cuestiones	-

Tabla 15 - CDU 7 - Crear versión base de documento

5.3.1.2.8 CDU 8 – Gestionar documentos de requisitos

Caso de Uso	**CDU 8 – Gestionar documento de requisitos**
Descripción	Operaciones de creación, consulta, edición y borrado de documentos.
Actores	Moderador
Otros actores	Responsable de Equipo y Analista
Asunciones	• El Moderador está autenticado en el sistema.
Pasos	1. El Moderador accede al repositorio. 2. El Moderador pulsa sobre la opción “Projects” 3. El Sistema proporciona un listado de proyectos sobre lo que el moderador tiene acceso. 4. El Moderador selecciona un proyecto p de los que tiene disponibles. 5. El Sistema muestra los documentos del proyecto. 6. El Moderador selecciona la opción de “Add document”. 7. El Moderador selecciona el tipo de documento. 8. El Moderador le asigna un nombre. 9. El Sistema guarda el documento y lo asigna al proyecto.
Variaciones	6.a.1 Fin del caso de uso. 6.b.1 El Moderador selecciona la opción de editar documento. 6.b.2 El Sistema muestra un formulario de edición de el documento. 6.b.3 El Moderador modifica la información que quiera y pulsa sobre “Submit”. 6.b.4 El Sistema guarda los cambios. 6.b.5 Fin del caso de uso 6.c.1 El Moderador selecciona la opción de “Delete”. 6.c.2 El Sistema elimina los requisitos de ese documento. 6.c.3 El Sistema elimina el documento. 6.c.4 Fin del caso de uso
Req. No-funcionales	-
Cuestiones	-

Tabla 16 - CDU 8 - Gestionar documentos de requisitos

5.3.1.2.9 CDU 9 – Gestionar hilos de discusión

Caso de Uso	CDU 9 – Gestionar hilos de discusión
Descripción	Creación y modificación de hilos de discusión.
Actores	Moderador
Asunciones	• El Moderador está autenticado en el sistema.
Pasos	1. El Moderador accede al Repositorio 2. El Moderador selecciona la opción de “Proyectos” 3. El Sistema muestra un listado de proyectos sobre lo que tiene acceso. 4. El Moderador selecciona un proyecto p de los que tiene disponibles. 5. El Sistema muestra un listado de documentos y requisitos sin clasificar en documentos. 6. El Moderador selecciona un documento d de los disponibles en p. 7. El Moderador selecciona un requisito r del documento d. 8. El Moderador selecciona la opción de crear hilo de discusión. 9. El Moderador establece la información del hilo, nombre y descripción. 10. El Sistema guarda el hilo en estado ABIERTO.
Variaciones	6.a.1 El Moderador selecciona un requisito sin clasificar. 5.a.1 El Moderador pulsa sobre la opción “Hilos de discusión” 5.a.2 El Sistema muestra un listado de todos los hilos de discusión del proyecto. 5.a.3 El Moderador selecciona un hilo y pulsa “Modificar”. 5.a.4 El Sistema abre el formulario del hilo en edición. 5.a.5 El Moderador modifica los datos que quiera y pulsa sobre “Aceptar”. 5.a.6 El Sistema guarda los cambios. 5.a.7 Fin del caso de uso 5.a.3.a.1 El Moderador selecciona un hilo. 5.a.3.a.2 El Sistema muestra los mensajes del hilo. 5.a.3.a.3 El Moderador selecciona un mensaje y pulsa eliminar. 5.a.3.a.4 El Sistema elimina el mensaje. 5.a.3.a.5 Fin del caso de uso.
Req. No-funcionales	-
Cuestiones	-

Tabla 17 - CDU 9 - Gestionar hilos de disución

5.3.1.2.10 CDU 10 - Gestionar requisitos

Caso de Uso	CDU 11 – Gestionar requisitos
Descripción	El Representante de Equipo/Analista añade requisitos a un documento de un proyecto.
Actores	Representante de Equipo/Analista
Otros actores	Responsable del Catálogo (mismas operaciones pero sobre un catálogo)
Asunciones	• El Representante de Equipo/Analista está autenticado en el sistema.
Pasos	1. El Representante de Equipo/Analista accede al repositorio. 2. El Representante de Equipo/Analista pulsa sobre la opción “Projects”. 3. El Sistema muestra un listado de proyectos sobre los que se tiene acceso. 4. El Representante de Equipo/Analista selecciona un proyecto p de los que tiene disponibles. 5. El Representante de Equipo/Analista selecciona un documento d de los disponibles en p. 6. El Representante de Equipo/Analista selecciona la opción de añadir requisito. 7. El Representante de Equipo/Analista introduce el texto del requisito, rellena los campos deseados y pulsa “Submit”. 8. El Sistema crea el nuevo requisito.
Variaciones	5.a.1 El Representante de Equipo/Analista pulsa sobre “add requirement”. 5.a.2 (continuar por 7) 5.b.1 El Representante de Equipo/Analista selecciona un requisito del listado de requisitos sin asignar a documento y pulsa sobre eliminar. 5.b.2 El Sistema elimina el requisito. 5.b.3 Fin del caso de uso. 5.c.1 El Representante de Equipo/Analista selecciona un requisito del listado de requisitos sin asignar a documento y pulsa sobre modificar. 5.c.2 El Sistema abre en edición el requisitos. 5.c.3 El Representante de Equipo/Analista modifica los datos que quiera y pulsa sobre “Submit”. 5.c.4 El Sistema guarda los cambios. 5.c.5 Fin del caso de uso. 5.d.1 El Representante de Equipo/Analista selecciona un requisito del listado de requisitos sin asignar a documento y pulsa sobre modificar. 5.d.2 El Sistema muestra las trazas del requisito. 5.d.3 El Representante de Equipo/Analista selecciona una traza y pulsa sobre “créate new trace”. 5.d.3 El Representante de Equipo/Analista selecciona el tipo de traza y el requisito destino y pulsa “Submit”. 5.d.4 El Sistema crea la traza.

	5.d.5 Fin del caso de uso. 5.d.3.a.1 El Representante de Equipo/Analista pulsa sobre eliminar traza. 5.d.3.a.2 El Sistema elimina la traza. 5.d.3.a.3 Fin del caso de uso. 6.a.1 El Representante de Equipo/Analista selecciona un requisito y pulsa sobre eliminar. 6.a.2 El Sistema elimina el requisito. 6.a.3 Fin del caso de uso. 6.b.1 El Representante de Equipo/Analista selecciona un requisito y pulsa sobre modificar. 6.b.2 El Sistema abre en edición el requisitos. 6.b.3 El Representante de Equipo/Analista modifica los datos que quiera y pulsa sobre "Submit". 6.b.4 El Sistema guarda los cambios. 6.b.5 Fin del caso de uso. 6.c.1 El Representante de Equipo/Analista selecciona un requisito. 6.c.2 El Sistema muestra las trazas del requisito. 6.c.3 El Representante de Equipo/Analista selecciona una traza y pulsa sobre "Create new trace". 6.c.3 El Representante de Equipo/Analista selecciona el tipo de traza y el requisito destino y pulsa "Aceptar". 6.c.4 El Sistema crea la traza. 6.c.5 Fin del caso de uso. 6.c.3.a.1 El Representante de Equipo/Analista pulsa sobre eliminar traza. 6.c.3.a.2 El Sistema elimina la traza. 6.c.3.a.3 Fin del caso de uso. 7.a.1 El Representante de Equipo/Analista introduce el texto del requisito y los parámetros entre "[]", rellena los campos deseados y pulsa "Submit".
Req. No-funcionales	-
Cuestiones	-

Tabla 18 - CDU 10 - Gestionar requisitos

5.3.1.2.11 CDU 11 – Gestionar secciones de documentos

Caso de Uso	CDU 12 – Gestionar secciones de documentos
Descripción	El Representante de Equipo/Analista selecciona un catálogo de requisitos y lo aplica al proyecto.
Actores	Representante de Equipo/Analista
Otros actores	Responsable del repositorio (mismas operaciones sobre catálogos).
Asunciones	• El Representante de Equipo/Analista está autenticado en el sistema.
Pasos	1. El Representante de Equipo/Analista accede al repositorio. 2. El Representante de Equipo/Analista pulsa sobre la opción “Projects”. 3. El Sistema muestra un listado de proyectos sobre los que se tiene acceso. 4. El Representante de Equipo/Analista selecciona un proyecto p de los que tiene disponibles. 5. El Representante de Equipo/Analista selecciona un documento d de p. 6. El Representante de Equipo/Analista selecciona la opción crear nueva sección. 7. El Representante de Equipo/Analista introduce el nombre de la sección. 8. El Sistema crea la nueva sección del documento.
Variaciones	6.a.1 El Representante de Equipo/Analista selecciona una sección y pulsa sobre “Edit”. 6.a.2 El Sistema muestra en edición la información de la sección. 6.a.3 El Representante de Equipo/Analista modifica la información que quiera y pulsa “Submit”. 6.a.4 El Sistema guarda los cambios. 6.a.5 Fin del caso de uso. 6.b.1 El Representante de Equipo/Analista selecciona una sección y pulsa sobre “Delete”. 6.b.2 El Sistema elimina todos los requisitos de la sección. 6.b.3 El Sistema elimina la sección. 6.b.4 Fin del caso de uso. 6.c.1 El Representante de Equipo / Analista selecciona la opción “Manage requirements” 6.c.2 Se activa el caso de uso CDU – 11 (Gestionar requisitos) 6.c.3 Fin del caso de uso
Req. No-funcionales	-
Cuestiones	-

Tabla 19 - CDU 11 - Gestionar secciones de documentos

5.3.1.2.12 CDU 12 – Participar en hilos de discusión

Caso de Uso	CDU 13 – Participar en hilos de discusión
Descripción	Consultar los hilos de discusión de un proyecto y enviar mensajes a los hilos abiertos.
Actores	Representante de Equipo/Analista
Asunciones	• El Representante de Equipo/Analista está autenticado en el sistema. • Existe un proyecto p al que pertenece el Representante de Equipo/Analista. • Existe al menos un documento d perteneciente a p. • Existe al menos un requisito r en el documento d. • El requisito r tiene al menos un hilo h abierto.
Pasos	1. El Representante de Equipo/Analista accede al repositorio 2. El Representante de Equipo/Analista pulsa sobre la opción “Projects”. 3. El Sistema muestra un listado de proyectos sobre lo que se tiene acceso. 4. El Representante de Equipo/Analista selecciona un proyecto p de los que tiene disponibles. 5. El Sistema muestra un listado de hilos de discusión disponibles. 6. El Representante de Equipo/Analista selecciona un hilo de discusión abierto. 7. El Representante de Equipo/Analista selecciona la opción de enviar mensaje. 8. El Representante de Equipo/Analista redacta el mensaje y lo envía. 9. El Sistema guarda el mensaje en el hilo.
Variaciones	6.a.1 El Representante de Equipo/Analista selecciona un hilo cualquiera. 6.a.2 El Sistema muestra los mensajes del hilos. 6.a.3 Fin del caso de uso.
Req. No-funcionales	-
Cuestiones	-

Tabla 20 - CDU 12 - Participar en hilos de discusión

5.3.1.2.13 CDU 13 – Usar un catálogo de requisitos

Caso de Uso	CDU 14 – Usar un catálogo de requisitos
Descripción	El Representante de Equipo/Analista reutiliza los requisitos de un catálogo en su proyecto.
Actores	Representante de Equipo/Analista
Asunciones	• El Representante de Equipo/Analista está autenticado en el sistema.
Pasos	1. El Representante de Equipo/Analista accede al repositorio. 2. El Representante de Equipo/Analista selecciona la opción "Projects". 3. El Sistema muestra un listado de proyectos sobre los que se tiene acceso. 4. El Representante de Equipo/Analista selecciona un proyecto p de los que tiene disponibles. 5. El Representante de Equipo/Analista la opción de aplicar catálogo. 6. El Representante de Equipo/Analista selecciona un catálogo de los disponibles. 7. Por cada documento: a. Si el tipo de documento no existe i. Se crea el tipo de documento ii. Se copian los requisitos b. Si el tipo de documento si existe i. Se copian los requisitos y se les cambia el orden del identificador por el siguiente que toque. c. Se copian todas las trazas d. Se copian todos los requisitos que no tienen documento asociado.
Variaciones	-
Req. No-funcionales	-
Cuestiones	-

Tabla 21 - CDU 13 - Usar un catálogo de requisitos

5.3.1.2.14 CDU 14 – Crear parámetro

Caso de Uso	CDU 15 – Crear parámetro
Descripción	El Representante de Equipo/Analista crea valores de parámetros para tipos de parámetros.
Actores	Representante de Equipo/Analista
Asunciones	• El Representante de Equipo/Analista está autenticado en el sistema.
Pasos	1. El Representante de Equipo/Analista accede al repositorio. 2. El Representante de Equipo/Analista selecciona la opción “Param types”. 3. El Sistema muestra un listado tipos de parámetros. 4. El Representante de Equipo/Analista selecciona un tipo de parámetro. 5. El Sistema muestra un listado de valores para ese tipo. 6. El Representante de Equipo/Analista pulsa sobre “add param value”. 7. El Sistema proporciona un formulario de creación de parámetros con el campo nombre y descripción. 8. El Representante de Equipo/Analista rellena el formulario y pulsa “Submit”. 9. El Sistema crea el nuevo valor del parámetro.
Variaciones	-
Req. No-funcionales	-
Cuestiones	-

Tabla 22 - CDU 14 - Crear parámetro

5.3.1.2.15 CDU 15 – Consultar proyecto

Caso de Uso	CDU 16 – Consultar proyecto
Descripción	El Usuario / Usuario Clave visualizar los requisitos de los proyectos a los que pertenece.
Actores	Usuario / Usuario clave
Asunciones	• El Usuario / Usuario clave está autenticado en el sistema.
Pasos	1. El Usuario / Usuario accede al repositorio. 2. El Usuario / Usuario selecciona la opción “Projects”. 3. El Sistema muestra un listado de proyectos sobre los que se tiene acceso. 4. El Usuario / Usuario selecciona un proyecto y visualiza sus datos, documentos, secciones y requisitos.
Variaciones	-
Req. No-funcionales	-
Cuestiones	-

Tabla 23 - CDU 15 - Consultar un proyecto

5.3.1.2.16 CDU 16 – Gestionar patrones de requisitos

Caso de Uso	CDU 16 – Gestionar patrones de requisitos
Descripción	El Responsable del repositorio gestiona las patrones de requisitos existentes en el repositorio.
Actores	Responsable del repositorio
Asunciones	• El Responsable del repositorio está autenticado en el sistema.
Pasos	1. El Responsable del repositorio pulsa sobre la opción "Patterns". 2. El sistema muestra un listado de patrones disponibles. 3. El Responsable del repositorio selecciona aqueelos que quiera elimina y pulsa sobre "Submit".
Variaciones	3.a El Responsable del repositorio pulsa sobre el botón de subir un fichero. 3.b El sistema muestra una ventana para cargar un fichero XML con el patrón implementado. 3.c El Responsable del repositorio pulsa sobre el botón de "Submit". 3.d El sistema guarda el patrón.
Req. No-funcionales	-
Cuestiones	-

5.3.2 Módulo PANTALASA_CMS

Cómo hemos visto en la sección de "Análisis y selección de alternativas", Drupal es un CMS que permite ampliar su funcionalidad a través de módulos.

En muchas ocasiones la mejor solución para ciertas necesidades es escribir un módulo personalizado; y más aún, cuando aquellas son demasiado específicas como para esperar que hayan sido tratadas o resueltas por módulos contribuidos.

Un módulo es un conjunto de archivos de programa, imágenes, datos, hojas de estilo, etc., organizados en un directorio, cuyo propósito es realizar tareas ligadas con la solución de una necesidad o de un grupo de necesidades afines. [19]

Un módulo tiene una estructura sencilla compuesta básicamente por un nombre y dos ficheros. El nombre debe ser simple y no debe empezar por un dígito ni incluir guiones. Los dos ficheros son los ficheros .info y .module.

El fichero .info contiene información de declaración del módulo y su objetivo es el de dar información sobre nuestro módulo al motor de Drupal. Ofrece principalmente información sobre el nombre del módulo, una descripción, el paquete, las dependencias con otros módulos, ficheros de los que se compone y core de Drupal para el que está disponible.

El nombre del fichero .info ha de ser "nombre_modulo.info".

El fichero .module, cuyo nombre ha de ser "nombre_modulo.module" es un script en lenguaje PHP que ofrece las funciones necesarias para que el módulo sea operativo. Aunque sea un script PHP, por convención, no termina con la cadena "?>".

De forma opcional y dependiendo de las necesidades, complejidad y funcionalidad que queramos ofrecer con nuestro módulo podemos crear una estructura de directorios y ficheros más compleja e incluir los ficheros en el fichero .module como include y/o require como ha de hacerse en PHP.

Adicionalmente podemos incluir un fichero .install que incluir funciones que sólo se quieren ejecutar una vez como puede ser crear o eliminar algún tipo de contenido.

Para el desarrollo de nuestra herramienta hemos decido crear un módulo personalizado por varios motivos: la funcionalidad que se pretende ofrecer es muy compleja para desarrollarla solamente con las opciones que ofrece Drupal en el core, haciendo hincapié en la gestión de permisos de los roles de usuario y distinciones entre usuarios de un mismo rol, facilitar la instalación de la herramienta y permitir futuras ampliaciones y modificaciones de la herramienta.

Nuestro módulo tendrá como nombre “pantalasa_cms” y definiremos tres ficheros principales: info, module e install.

5.3.2.1 Decisiones de diseño

Vamos a comentar las decisiones tomadas a la hora de diseñar la herramienta con las opciones y la funcionalidad que nos ofrece Drupal.

Por un lado tendremos el contenido propiamente dicho almacenado con metainformación asociada. Estos son los tipos de contenidos que se van a crear y van a tener persistencia en la herramienta. Además también tenemos vocabularios de términos.

Esos contenidos habrán de representarse gráficamente para que puedan ser manejados de forma sencilla y eficiente por los usuarios de la herramienta. Para ello contamos con las vistas.

También habrá que proporcionar funcionalidad con esos tipos de contenido para poder crearlos y manipularlos según las especificaciones de la herramienta.

Finalmente debemos limitar la funcionalidad y las opciones disponibles a según qué usuarios representados por roles.

Así que a modo general, el esquema de nuestra herramienta sería el siguiente:

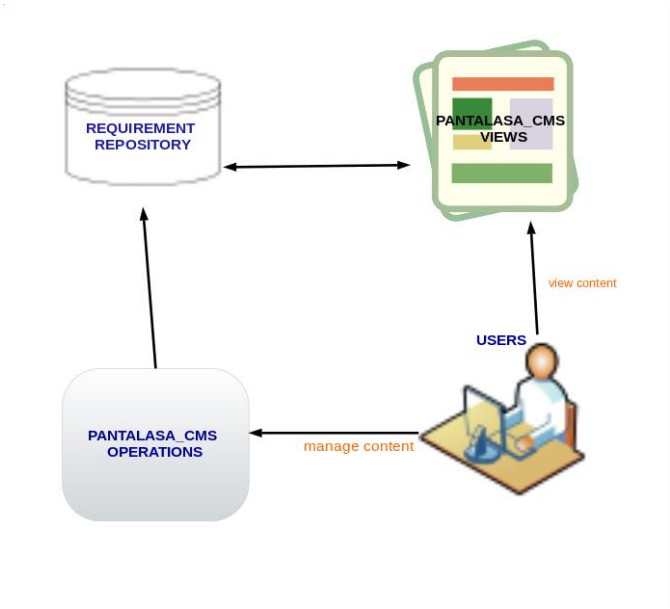


Figura 13 - Esquema PANTALSA-CMS (simple)

Con las opciones que nos proporciona Drupal y los diferentes módulos que vamos a usar, como el módulo views, no podemos implementar toda la funcionalidad de la herramienta. Si bien es cierto que podemos personalizar los permisos para limitar el acceso a según qué roles pero no podemos hacer distinciones complejas entre usuarios de un mismo rol. Por ejemplo, un usuario analista ha de tener permiso para crear un documento en un proyecto pero no en todos los proyectos. Esto no se puede restringir con la gestión de permisos que nos proporciona Drupal **de modo que hemos de implementarlo.**

Del mismo modo ocurre con las vistas. El caso anterior si es posible implementarlo con el módulo views pero cuando se trata de vistas más complejas nos vemos de nuevo limitados en cuanto a funcionalidad. **También hemos de implementar el acceso a las vistas para tener un control más exaustivo a la hora de acceder a una vista.**

Así pues, el esquema anterior queda ampliado de la siguiente forma:

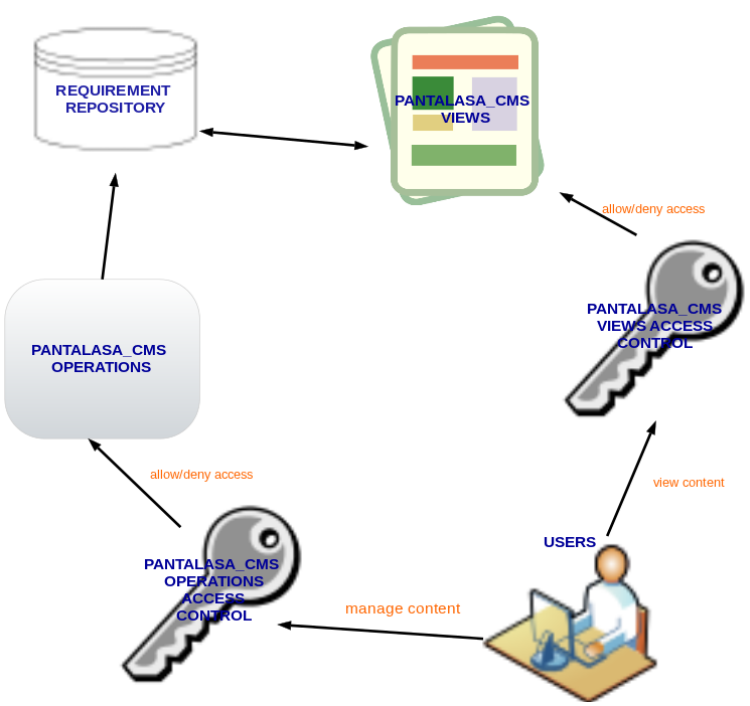


Figura 14 - Esquema PANTALSA-CMS (completo)

5.3.2.1.1 Tipos de contenido

A continuación se detallan los objetos de dominio que van a tener representación en la herramienta como tipos de contenido en Drupal:

- Catálogo
- Proyecto
- Documento
- Sección
- Requisito
- Tipo de parámetro
- Valor de parámetro
- Instancia de parámetro
- Hilo de discusión
- Traza
- Usuario
- Grupo

Estos tipos de contenido han sido extraidos del diagrama conceptual que podemos ver en la figura Figura 15:

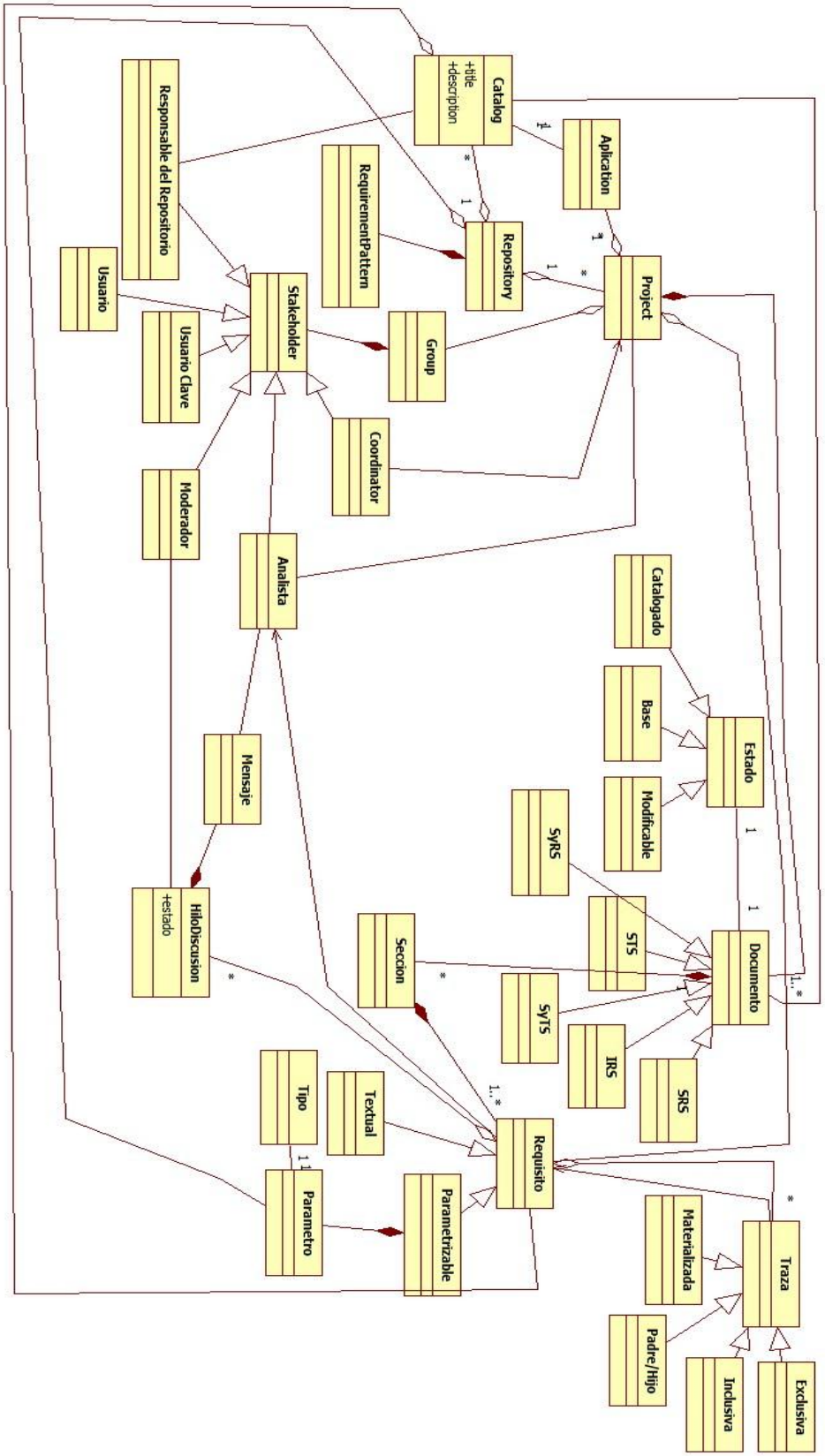


Figura 15 - Diagrama conceptual

5.3.2.1.1.1 Catálogo

Un catálogo representa una colección de requisitos que pueden estar, o no, organizados en documentos. Un catálogo tiene un nombre indicando que es de un perfil o dominio y una descripción para que sirva de ayuda a los usuarios.

Debemos representar los requisitos que no están organizados en documentos y que cuelga directamente del catálogo. Es una colección de 0 o más elementos de tipo requisito.

Se toma la decisión de no crear una relación de 0 a muchos desde catálogo hasta documento de forma que no vamos a indicar que un catálogo tiene varios documentos sino que un documento pertenece a un catálogo (o proyecto). La razón de tomar esta decisión es debido a la funcionalidad que se va a implementar y a cómo controlar los accesos de los usuarios de forma eficiente. La mayoría de las operaciones que se van a llevar a cabo son a nivel de documento, por tanto, es más eficiente recuperar un catálogo desde el propio documento que tener que navegar por todos los catálogos disponibles para recuperar un documento y sus atributos.

5.3.2.1.1.2 Proyecto

Un proyecto es muy similar a un catálogo. Tiene un título y una descripción breve y una colección de requisitos sin clasificar en documentos. Las razones para no referenciar a los documentos de un proyecto desde el propio proyecto son las mismas que se han dado en el punto anterior.

Además un proyecto tiene un coordinador y un conjunto de usuarios pero desde el punto de vista funcional, es más útil decir que un usuario pertenece a un proyecto que un proyecto tenga varios usuarios. De esta forma podemos representar que un usuario pueda estar en varios proyectos de forma sencilla y eficiente.

5.3.2.1.1.3 Documento

Un documento es una colección de requisitos y secciones de requisitos. Un documento tiene un nombre y una descripción.

Como hemos dicho antes, un documento referencia al proyecto o catálogo que lo contiene para, de forma sencilla y rápida, poder recuperar dicho proyecto o catálogo y comprobar los permisos que se tiene sobre él. Así pues, aquí se establece la referencia 1 a 1 de un documento a un catálogo o proyecto debido a que Drupal permite referenciar a varios tipos de contenido con un mismo campo.

Tenemos dos atributos más que requieren representación. Estos son el tipo de documento y el estado del documento, que no son más que dos referencias a los vocabularios que contienen los valores posibles de términos para estos atributos.

Por último, se establecer una relación 0 a muchos con el tipo de contenido requisito para representar los requisitos que no están organizados en secciones.

5.3.2.1.1.4 Sección

Las secciones de un documento son colecciones de requisitos pero no vamos a crear una referencia 0 a muchos desde sección hasta requisito para representar los requisitos que contiene una sección sino porque, por lo general, los requisitos se estructuran dentro de secciones y para evitar una explosión de relaciones sección-requisito vamos a referenciar la sección desde el tipo requisito.

Además, una sección puede contener varias subsecciones donde cada una de ellas tiene los mismos atributos. No vamos a referenciar a las subsecciones sino que será cada sección la que referencie a la sección contenedora, si existe. Este atributo será opcional.

Debido a la relación entre secciones hemos de incluir un número de sección que será de tipo entero, o mejor dicho, un número natural.

Se incluye además un título de la sección y una breve descripción de forma opcional.

5.3.2.1.1.5 Requisito

Un requisito es una sentencia con un conjunto de metadatos asociados. La parte principal es el texto del requisito y un nombre identificativo. Sin embargo, no basta con un nombre, de modo que hemos de añadir un identificador único de requisito. El formato de este identificador será el siguiente:

[P|C]-[NC|[*TIPO_DOC*]]-[*NID*]

P indica que pertenece a un Proyecto.

C indica que pertenece a un Catálogo.

NC indica que no está organizado en ningún documento.

TIPO_COD son las siglas del tipo de documento.

NID es el node id que le asigna Drupal.

De esta forma nos aseguramos que el identificador sea único y manejable de forma que rápidamente podemos averiguar dónde está un requisito dentro de un catálogo o proyecto.

Un requisito tiene una relación opcional con una sección para indicar que pertenece a una sección concreta. Esta relación se implementa en este sentido debido a que a la hora de trabajar con requisitos podemos rápidamente averiguar a qué sección pertenece, y esa sección, a qué documento pertenece, y por consiguiente, a qué proyecto o catálogo pertenece.

Un requisito también tiene una serie de atributos con valores restringidos como son el riesgo, criticidad, estado, prioridad, cumplimiento y método de verificación los cuales relacionaremos con los vocabularios correspondientes.

Otros campos son la motivación del requisito y la fuente del requisito.

También consta de referencias opcionales al usuario que lo propuso, al analista fuente y a los grupos de trabajo fuente y destino.

5.3.2.1.1.6 Tipo de parámetro

Los requisitos parametrizables tienen uno o más valores que pueden ser instanciados. Esas instancias son de algún tipo y es lo que se llama tipo de parámetro.

Un tipo de parámetro no va a referenciar a ningún requisito ni valor sino que serán las instancias las que referencien a su tipo.

Como atributos tiene un nombre del tipo, una breve descripción y un valor clave para poder referenciar al tipo dentro del texto del requisito. Este campo clave se referenciará mediante corchetes dentro del texto: [<clave>].

5.3.2.1.1.7 Valor de parámetro

Un valor de parámetro es una instancia concreta de un tipo de parámetro por lo que hemos de referenciar al tipo de parámetro al que pertenece.

Se añadirá un atributo título con el nombre del valor del parámetro y una descripción breve que sirva como ayuda.

5.3.2.1.1.8 Instancia de parámetro

Como hemos dicho, los requisitos parametrizables tienen en el campo texto referencias a tipos de parámetros mediante corchetes ([ ]) donde en medio se coloca el valor clave. Pero hemos de representar esta relación de alguna forma. Por eso se crea este tipo de contenido, para representar dicha relación.

Una instancia de parámetro referencia al requisito concreto, al tipo de parámetro del texto del requisito y al valor del parámetro.

5.3.2.1.1.9 Hilo de discusión

Un hilo de discusión es una colección de comentarios sobre un aspecto de un requisito. Para inicializar un hilo se requiere de un título del hilo, una descripción del motivo y el requisito al que hace alusión. Además, un hilo tiene un estado, abierto o cerrado.

Para los comentarios de un hilo haremos uso del sistema de comentarios que proporciona Drupal permitiendo que los usuarios añadan comentarios a este contenido en concreto.

5.3.2.1.1.10 Traza

Las trazas representan relaciones entre requisitos por lo que necesitamos referenciar tanto el requisito origen de la traza como el requisito con el que está relacionado, el requisito destino.

Además, necesitamos decir de qué tipo es la traza mediante un vocabulario donde están restringidos los valores.

En cuanto al nombre de la traza la usaremos para establecer un identificador numérico de traza basado en el siguiente número de la última traza del requisito y el identificador de requisito para poder identificarla de forma unívoca.

5.3.2.1.1.11 Usuario

El usuario está creado automáticamente por Drupal pero nos vamos a modificarlo para representar el conjunto de proyectos a los que pertenece un usuario.

Para el rol responsable no tiene sentido este campo ya que un responsable del repositorio no está asociado a ningún proyecto pero si para el resto de roles.

Para el rol coordinador, este campo representa los proyectos de los que es coordinador y por tanto tiene competencias. Para el resto de roles, representa los proyectos a lo que pertenece como miembro.

Este campo es vital para poder gestionar los permisos de forma correcta.

5.3.2.1.1.12 Grupo

Un grupo es una colección de usuarios en un proyecto concreto. Cada grupo tiene uno y solo un proyecto al que pertenece y los usuarios pueden pertenecer a uno o más grupos. Los usuarios, además, tiene que pertenecer al mismo proyecto que el grupo. Por tanto, necesitamos crear una referencia de 0 a muchos con los miembros del grupo que en esta versión de la herramienta sólo podrán ser analistas.

También tiene otra referencia obligatoria a un usuario concreto, el representante de equipo.

Además consta de un nombre y una descripción del grupo.

5.3.2.1.2 Creación de contenidos

Para la creación de contenidos Drupal proporciona una funcionalidad genérica con formularios estándar donde se presentan todos los campos y relaciones creados pero presenta varias limitaciones de control de acceso para usuarios y roles, de modo que hemos de buscar una alternativa a la forma de creación.

Drupal tiene una API Form para crear formularios personalizados donde podemos mostrar los campos que sean realmente necesarios y queramos que el usuario visualice y edite. Con esto nos aseguramos que el usuario manipula la información que queremos que manipule pero no podemos hacer nada con los permisos.

Estas comprobaciones de permisos y control de acceso se llevarán a cabo antes de proporcionar el formulario y serán detalladas más adelante.

5.3.2.1.3 Visualización de contenidos

Para mostrar y recuperar la información del repositorio y formatearla de forma agradable al usuario se hará uso del módulo views que de forma sencilla y visual permite crear vistas en forma de tablas, listados, etc.

Las vistas que vamos a crear son vistas genéricas, es decir, sin control de acceso ya que debido a la complejidad que presenta la gestión de permisos no es posible implementarlo con las vistas.

Implementaremos unas funciones de control de acceso para que se ejecuten antes de mostrar la vista. Una función interesante que tiene el módulo views, la cual vamos a aprovechar para

llevar a cabo nuestra implementación, es la función hook_ views_pre_render que recibe como parámetro una referencia a la vista. Lo que hacemos es redefinir esta función con nuestra propia implementación que se trata un control de casos en función del nombre de la vista. Dependiendo del nombre de la vista y su display ejecutaremos una u otra función.

Cada una de estas funciones hace un control de los permisos que se quieran controlar y en caso de no satisfacerse las condiciones se ejecutará una función que proporciona Drupal, drupal_access_denied(), que nos retorna inmediantamente a una página de acceso denegado. Si por el contrario se cumplen las condiciones necesarias se retornaría sin más y se ejecutaría la vista.

5.3.2.1.4 Exportar documentos

Es de utilidad poder exportar los tipos documentos con su estructura interna a un documento PDF, Word, etc.

Para no limitar ni imponer el formato de salida de un documento se ha optado por la generación de un documento XML con toda la información a partir un XML Schema creado de forma que el esquema de un documento sea estándar.

La idea es que se creen otros módulos de transformación de documentos XML al formato concreto basados en hojas de estilos XSL.

5.3.2.2 Nuestro módulo en Drupal

El esquema general de Drupal podemos verlo en la Figura 16:

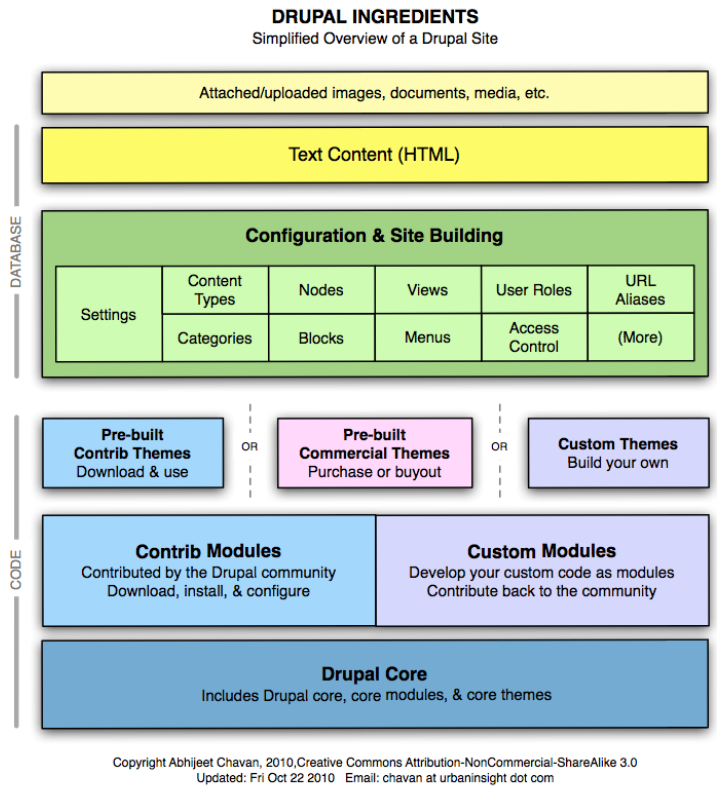


Figura 16 - Esquema de Drupal

Nuestro módulo, PANTALASA_CMS, irá dentro de "Custom Modules" y será el fichero "pantalasa_cms.install" quien ejecute acciones para añadir y modificar la base de datos de Drupal. En la Figura 17 podemos ver donde se ubica nuestro módulo:

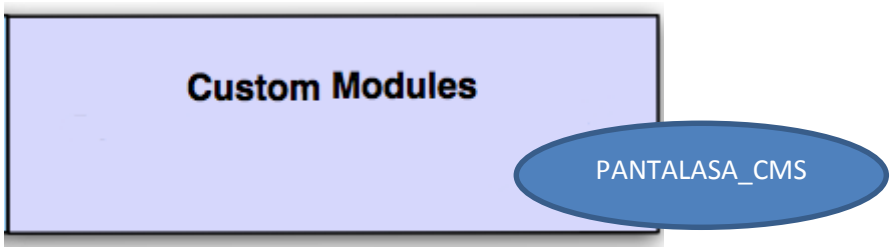


Figura 17 - PANTALASA-CMS como módulo

Su instalación es la que modifica la base de datos de Drupal. Esas modificaciones las realiza Drupal ya que el fichero .install invoca a la funcionalidad de Drupal para que este cree los tipos de contenido, menús, bloques, etc. En la Figura 18 hemos puesto esta relación fuera del esquema de Drupal para dejar claro que esas modificaciones las invoca el fichero .install:

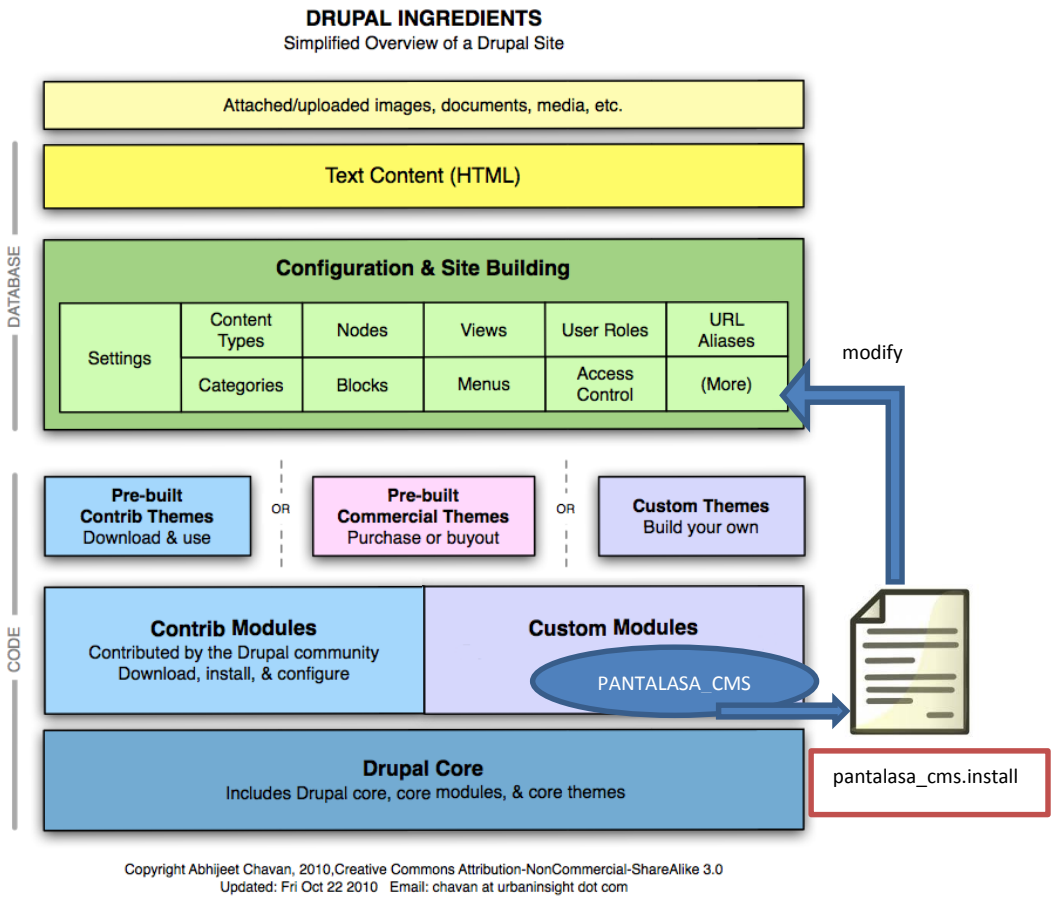


Figura 18 - Pantalasa CMS en Drupal

5.3.2.2.1 Esquema de ficheros y directorios

Antes de pasar a detallar los principales ficheros del módulo y sus características vamos a definir la estructura que se ha seguido para organizar los distintos tipos de ficheros según su tipo y funcionalidad. En el la Figura 19 se muestra un diagrama con la estructura de directorios:

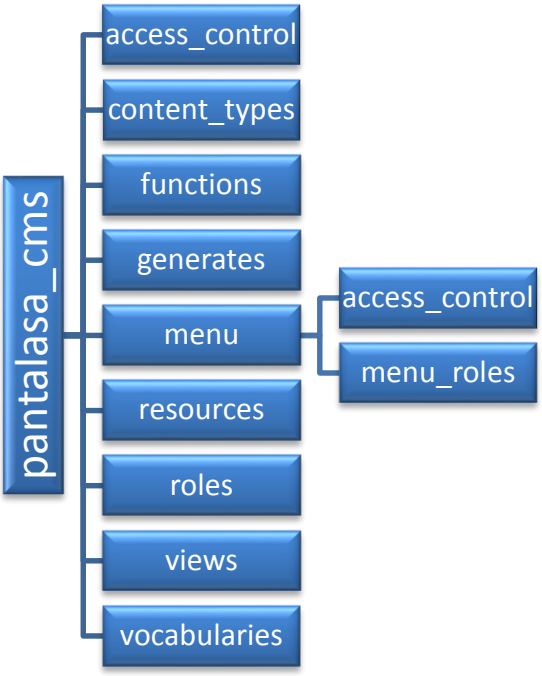


Figura 19 - Estructura de direcorios de Pantalasa CMS

A continuación vamos a describir en la Tabla 24 los tipos de ficheros que hay en cada directorio:

Directorio	Descripción
pantalasa_cms	Directorio principal del módulo. Contiene los ficheros pantalasa_cms.info pantalasa_cms.module y pantalasa_cms.install.
access_control	Control de acceso a las vistas del directorio views.
content_types	Definición e instalación de todos los tipos de contenido del módulo.
functions	Funciones genéricas de la herramienta.
generates	Directorio para guardar los ficheros XML generados para exportar documentos.
menu	Definición e instalación del menú de pantalasa_cms. (hook_menu)
menu/access_control	Control de acceso a las distintas operaciones del menú.
menu/menu_roles	Definición e instalación de los menús de cada tipo de usuario.
resources	Directorio de recursos.
Roles	Definición y creación de roles.
views	Definición de vistas del módulo views.

Directorio	Descripción
vocabularies	Definición e instalación de los vocabularios con sus términos.

Tabla 24 - Tipos de ficheros en directorios

5.3.2.2.2 Fichero pantalasa_cms.info

En nuestro fichero pantalasa-cms.info indicaremos la información que se muestra en la tal y como podemos encontrar en [20]:

Nombre	Valor
Name	Pantalasa CMS
Description	Requirements management in GSD
Package	Requirements management
Core	7.x
Dependencies (array)	{references, menú, node, views, views_ui, node_reference, user_reference, php}

Tabla 25 - Información fichero pantalasa_cms.info

Este fichero no tiene nada más de interés que sea necesario comentar.

5.3.2.2.3 Fichero pantalasa_cms.install

El fichero pantalasa_cms.install tiene funciones que se ejecutan la primera vez que se habilita el módulo y también cuando se desinstala. La función a implementar que se ejecutará al inicio en el hook_install como pantalasa_cms _install y para desinstalar el hook_uninstall como pantalasa_cms _uninstall.

5.3.2.2.3.1 Función pantalasa_cms _install

La función pantalasa_cms _install se encarga de crear los tipos de contenido, vocabularios, roles, bloques, vistas y establecer los permisos necesarios para la herramienta. Lo hace dividiendo el código en varias funciones. A continuación mostramos el árbol de llamadas que se genera cuando Drupal invoca a pantalasa_cms_install():

1. pantalasa_cms _crear_vocabularios()
2. pantalasa_cms _crear_content_types()
3. pantalasa_cms _crear_roles()
4. pantalasa_cms _actualizar_content_types_con_roles()
5. pantalasa_cms _add_user_fields()
6. pantalasa_cms _crear_menus()
7. pantalasa_cms _actualizar_menus()
8. pantalasa_cms _crear_content()

Debemos indicar que entre las llamadas de pantalasa_cms _crear_roles() y pantalasa_cms _actualizar_content_types_con_roles() y entre las llamadas pantalasa_cms _actualizar_menus() y pantalasa_cms _crear_content() hemos de limpiar la caché para que los datos que van a manejar las segundas funciones (pantalasa_cms _actualizar_content_types_con_roles() y pantalasa_cms _crear_content()) tenga información consistente para ejecutarse y llevar a cabo su cometido.

A continuación vamos a detallar sólamente qué hace cada función. Los motivos de por qué hacemos cada cosa en cada función son decisiones de diseño que se explicarán más adelante:

5.3.2.2.3.2 Función pantalasa_cms _crear_vocabularios

Esta función accede al directorio dentro de nuestro módulo que contiene los ficheros con el código de creación de los vocabularios e incluye el código de esos ficheros.

5.3.2.2.3.3 Función pantalasa_cms _crear_content_types

Esta función accede al directorio dentro de nuestro módulo que contiene los ficheros con el código de creación de los diferentes tipos de contenido e incluye el código de esos ficheros.

5.3.2.2.3.4 Función pantalasa_cms _actualizar_content_types_con_roles

Existen tipos de contenido que contienen campos que son referencias a usuarios de uno o varios roles determinados. Más adelante se detallarán los distintos tipos de contenido y sus campos.

En esta función se actualizan esos campos que son referencias a usuarios ya que en el momento de creación del tipo de contenido aún no se han creado los roles y no se puede establecer qué roles son referenciables por un campo y hemos de diferir el establecimiento de esta restricción a después de haber creado los tipos de contenido y los roles.

Se cargan aquí los distintos campos y se establece dentro de su arreglo el valor del rid (role id) del role correspondiente.

```
$field['settings']['referenceable_roles'][roleId]=roleId;
```

Finalmente se actualiza el campo:

```
field_update_field($field);
```

5.3.2.2.3.5 Función pantalasa_cms _add_user_fields

Esta función, al igual que la anterior, también modifica un campo pero en este caso de un usuario. Hemos querido implementar esta funcionalidad en una función separada de la anterior debido a que se trata de un usuario, no de un tipo de contenido, y además ese usuario es creado por Drupal y no por PANTALASA_CMS.

Se añade un campo que referencia al tipo de contenido que representa un proyecto para poder representar que un usuario pertenece a ninguno, a uno o a varios proyectos.

5.3.2.2.3.6 Función pantalasa_cms _crear_menus

Esta función accede al directorio dentro de nuestro módulo que contiene los ficheros con el código de creación de los diferentes menús de navegación e incluye el código de esos ficheros.

5.3.2.2.3.7 Función pantalasa_cms _actualizar_menus

Los menús de navegación se representan como tipo menú en Drupal a través del módulo que incluye en el core, Menú. Pero para que tengan representación visual en un lugar de la página web necesitan estar contenidos dentro de un bloque, creado con el módulo Block de Drupal.

Cuando se crea un menú también se crea su bloque pero por defecto no se activa. Con esta función lo que hacemos es activar los bloque de menú y establecemos los permisos de cada bloque y en qué páginas se mostrará el bloque dependiendo del tipo de role que sea.

Usamos una función propia llamada pantalasa_cms _activate_block para activar el bloque y para asignar permisos al bloque lo hacemos a través del API para la base de datos que ofrece Drupal.

Con la función pantalasa_cms _activate_block establecemos en qué región se mostará el bloque, en qué páginas se mostrará y para qué themes se mostrará.

Para establecer las restricciones de roles debemos insertar en la tabla 'block_role' los valores del módulo que ha creado el bloque, el nombre del bloque y el role id del role que tiene permisos para visualizarlo.

5.3.2.2.3.8 Función pantalasa_cms _crear_content

Finalmente se ejecuta esta función cuyo propósito es crear el contenido necesario para empezar a usar la herramienta.

Se crea automáticamente el usuario 'Responsable' que tiene el role de responsable del repositorio y también se crea un nodo de tipo 'Page' y se establece como página de inicio de la herramienta.

5.3.2.2.3.9 Función pantalasa_cms _uninstall

Cuando decidimos elimina nuestro módulo hemos de llevar a cabo una serie de tareas donde se eliminarán todos los tipos de contenido, roles, menús, vistas y vocabularios creados.

La secuencia de llamadas que se ejecutan cuando Drupal invoca a la función pantalasa_cms_uninstall es la siguiente:

1. pantalasa_cms _eliminar_vocabularios()
2. pantalasa_cms _eliminar_content_types()
3. pantalasa_cms _eliminar_roles()
4. pantalasa_cms _eliminar_menus()
5. pantalasa_cms _eliminar_vistas()
6. pantalasa_cms _eliminar_content()

5.3.2.2.3.10 Función pantalasa_cms_eliminar_vocabularios

Los vocabularios necesarios para PANTALASA_CMS son creados a través del módulo taxonomy. Este módulo tiene una función llamada taxonomy_vocabulary_delete() que recibe como parámetro el identificador único del vocabulario. Una forma sencilla de recuperar el objeto vocabulario es a través de la función taxonomy_vocabulary_machine_name_load() que recibe como parámetro el nombre de máquina del vocabulario. [21]

En esta función creamos un array con el nombre de todos los vocabularios el cual recorreremos para eliminar uno a uno los vocabularios.

5.3.2.2.3.11 Función pantalasa_cms_eliminar_content_types

Con esta función eliminaremos los tipos de contenido y sus campos creados.

En primer lugar eliminaremos todos los campos con la función field_delete_field() que recibe como parámetro el nombre máquina del campo a eliminar. [22]

Una vez se han eliminado los campos procedemos a eliminar los tipos de contenido con la función node_type_delete() del módulo node [23] que recibe como parámetro el nombre máquina del tipo de contenido a eliminar.

5.3.2.2.3.12 Función pantalasa_cms_eliminar_roles

Para eliminar los roles creados disponemos en el módulo user [24] de la función user_role_delete() que recibe como parámetro el nombre máquina del role a eliminar.

5.3.2.2.3.13 Función pantalasa_cms_eliminar_vistas

Las vistas creadas con el módulo views [25] se eliminan con esta función. A partir de un array que contiene los nombres máquina de las vistas se hace un recorrido por dicho array y se recupera el objeto de la vista.

Ese objeto tiene un método delete() que la elimina.

5.3.2.2.3.14 Función pantalasa_cms_eliminar_menus

Los menús de navegación se eliminan con esta función que tiene un array con los nombres máquina de todos los menús a eliminar y ejecuta la función menu_delete() que recibe como parámetro un objeto menú que se recupera con la función menu_load() del módulo 'Menu' [26]que recibe como parámetro el nombre máquina del menú.

5.3.2.2.3.15 Función pantalasa_cms_eliminar_content

Esta función es la encargada de eliminar el contenido creado en la instalación y que ya no sirve. Es el usuario 'Responsable'. Lo eliminamos con la función user_delete() del módulo 'User' que recibe como parámetro el user id del usuario.

Recuperamos el objeto usuario con la función user_load_by_name() que recibe como parámetro el nombre de usuario y devuelve un objeto de tipo user.

5.3.2.2.4 Fichero pantalasa_cms.module

Este es el fichero principal de nuestro módulo. En este fichero está implementada toda la funcionalidad que nuestro módulo ofrece. Consta de tres funciones principales que implementan el hook del módulo views y que comentaremos brevemente:

La función pantalasa_cms_views_api() es necesaria cuando se van a establecer vistas por defecto. Se indica la versión de la API.

La función pantalasa_cms_views_default_views() indica cuales son las vistas por defecto. En nuestro caso cargamos nuestras vistas que está implementadas en ficheros individuales dentro del directorio views.

La última función importante pantalasa_cms_views_pre_render() la cual se ejecuta antes de mostrar el resultado de cualquier vista. Más adelante se explicará el porqué de implementar esta función pero por ahora diremos que sirve para controlar los accesos a las vistas de forma individual por parte de los usuarios.

También incluye el fichero pantalasa_cms.module una serie de ficheros necesarios como funciones comunes, la implementación del hook_menu y los controles de acceso a vistas. Más adelante detallaremos todos estos ficheros y funciones.

5.3.3 Módulos adicionales para Drupal

Antes de pasar a detallar el diseño e implementación de la funcionalidad de la herramienta vamos a indicar los módulos que hemos usado y que son necesarios para la instalación y uso de la herramienta.

La distribución de Drupal viene con una serie de módulos preinstalados que ofrecen la funcionalidad básica para implementar un sitio web. Aunque puede ser posible realizar completamente la implementación de nuestra herramienta existen módulos que facilitan mucho esta tarea ya que están desarrollados y diseñados para realizar tareas específicas de una forma más eficiente que la que proporciona Drupal con esos módulos básicos.

Vamos a nombrar cada uno de los módulos usados junto con una descripción y vamos a indicar para qué hemos usado dicho módulo. En la Figura 20 podemos ver un diagrama con los módulos que se incluyen en Drupal y los módulos externos usados:

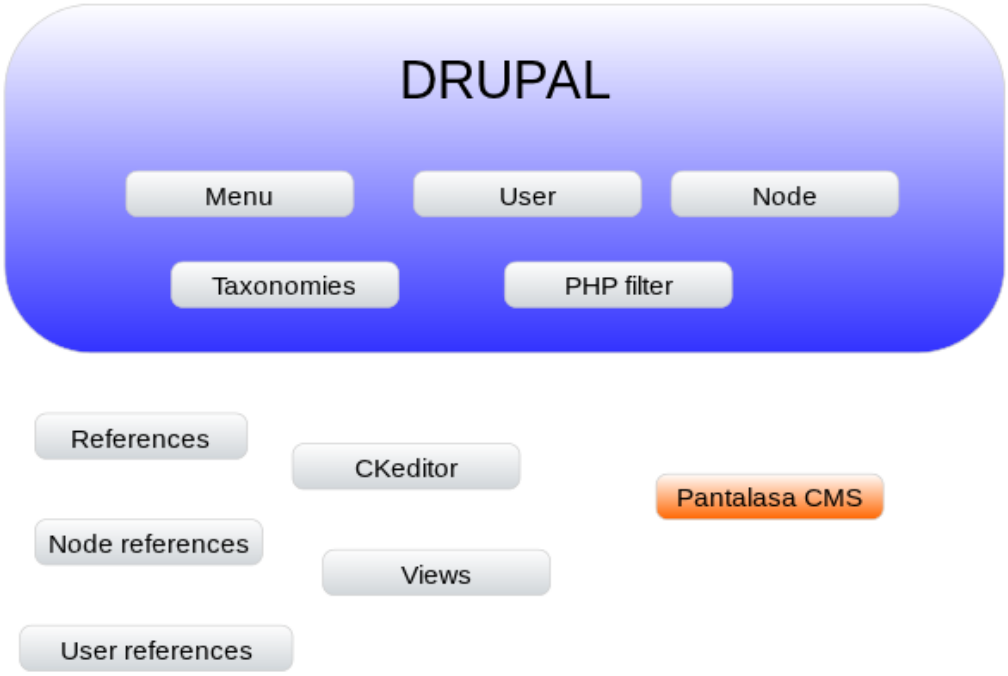


Figura 20 - Módulos necesarios para Pantalasa CMS

5.3.3.1 Módulo References, node_reference y user_reference

Proporciona referencias a nodos y usuarios de forma que es posible crear relaciones entre tipos de contenido. Lo podemos encontrar en el sitio web de Drupal. [27]

Los tipos de contenido que vamos a manejar tienen información básica como puede ser un nombre, una descripción o un atributo cualquiera con un formato de tipos básico como puede ser una cadena de texto o un número. Sin embargo, existen relaciones entre tipos de contenido como puede ser que un catálogo contenga requisitos, un requisito tiene un analista, etc.

Este módulo nos proporciona un tipo de campo referencia a un nodo o a un usuario de forma que podemos relacionar directamente un tipo de contenido con otro de forma simple.

5.3.3.2 Módulo views y views_ui

Con estos módulos podremos crear vistas de contenido muy personalizables tanto por los administradores como por los usuarios de la herramienta.

Queremos mostrar al usuario de forma clara y sencilla el contenido que existe estructurado como nodos en Drupal. Con estos módulos podemos crear visualmente y de forma rápida vistas muy avanzadas que el usuario podrá acceder a través de una URL para visualizar el contenido.

5.3.3.3 Módulo CKeditor

Este módulo permite insertar un editor de texto de tipo WYSIWYG (*What you see is what you get)* que será bastante últil a la hora de asignar secciones textuales a un documento de requisitos ya que podremos formaterlo como un editor de texto normal y obtener el código HTML.

5.4 Implementación del prototipo

5.4.1 Vocabularios y taxonomías

A la hora de clasificar distintos tipos de contenido en Drupal necesitamos hacer uso de algún tipo de palabra clave o identificador que nos indique claramente de qué tipo es un contenido cuando la variedad de tipos está limitada por el dominio.

Drupal trae incorporado en el core de su distribución un módulo llamado “Taxonomy” que permite la creación y gestión de vocabularios. [28]

Un vocabulario no es más que una colección de términos donde cada término tiene un significado que clasifica el objeto o tipo de contenido al que se aplique como es el caso de los dominios.

Los vocabularios tienen un nombre obligatorio y una descripción. Además, Drupal permite añadir nuevos campos para definir los términos de los que está formado. Por defecto un término sólo tiene un nombre y una descripción.

Haremos uso de vocabularios en la implementación de nuestra herramienta que se irán detallando a continuación.

Para crear un formulario tenemos la función pantalasa_cms_crear_vocabularios(). Esa función carga los ficheros que hay en el directorio /vocabularies/ donde cada fichero crea un vocabulario.

En cada fichero tenemos una estructura similar donde se define el vocabulario, se crea y se añaden los términos.

Para crear un vocabulario disponemos de la función taxonomy_vocabulary_save() del módulo "taxonomies" que recibe como parámetro un objeto vocabulario.

Ese objeto es implementado como un array con un nombre descriptivo, un nombre máquina y una descripción.

```
$vocabulary = (object) array(
      'name' => 'nombre descriptivo',
      'machine_name' => 'nombre_maquina',
      'description' => 'descripción del vocabulario',
);
taxonomy_term_save($vocabulary);
```

Una vez creado el vocabulario creamos los términos. Cada término es un array con un nombre, una descripción de término y un identificador de vocabulario.

```
$term = (object) array(
      'vid' => $vocabulary->vid,
      'name' => 'término',
      'description' => 'descripción del término',
);
taxonomy_term_save($term);
```

Para recuperar un vocabulario disponemos de la función "taxonomy_vocabulary_machine_name_load" del módulo "taxonomies" que recibe como parámetro el nombre máquina de un vocabulario y recupera un objeto vocabulario.

En la Tabla 26 indicaremos los vocabularios a crear y una descripción.

5.4.1.1 Tabla de vocabularios

Vocabulario	Nombre máquina	Descripción
Criticity	criticidad_requisito	Define los posibles valores para el atributo criticidad de un requisito.
Requirement compliance	cumplimiento_requisito	Define los posibles valores para el atributo cumplimiento de un requisito.
Requirement state	estado_requisito	Define los posibles valores para el atributo estado de un requisito.
Verification method	metodo_verif_requisito	Define los posibles valores para el atributo método de verificación de un requisito.
Requirement priority	prioridad_requisito	Define los posibles valores para el atributo prioridad de un requisito.
Requirement risk	riesgo_requisito	Define los posibles valores para el atributo riesgo de un requisito.
Document state	estado_documento	Define los posibles estados en lo que se puede encontrar un documento de requisitos.

Vocabulario	Nombre máquina	Descripción
Document type	tipo_documento	Define los posibles tipos de documentos.
Trace type	tipo_traza	Define los posibles tipos de traza entre requisitos.
Thread state	estado_hilo	Define los posibles estados de un hilo de discusión.

Tabla 26 - Tabla resumen de vocabularios

5.4.1.2 Tabla de términos

La muestra un resumen de los términos de cada vocabulario y una pequeña descripción:

Vocabulario	Término	Descripción
Document type (tipo_documento)	SRS	Especificación de Requisitos del Software
	SyRS	Especificación de Requisitos del Sistema
	STS	Especificación de Pruebas del Software
	SyTS	Especificación de Pruebas del Sistema
	IRS	Especificación de Requisitos de Interfaz
Document state (estado_documento)	Modifiable	En este estado se encuentran los documentos que se pueden modificar.
	Base	Representa una versión base de un documento. Los documentos en este estado no se pueden modificar.
	Listed	En este estado se encuentran los documentos de un catálogo. Pueden ser modificados solamente por el responsable del catálogo.
Thread state (estado_hilo)	Open	Se puede acceder al hilo de discusión y participar en él enviando mensajes.
	Closed	Se puede acceder al hilo pero no se puede participar. Sólo se puede consultar.
Trace type (tipo_traza)	Inclusive	Cuando se incluye el requisito origen se ha de incluir el destino.
	Exclusive	Cuando se incluye el requisito origen no se puede incluir el requisito destino.
	Father_son	Apunta al requisito del que procede.
	Related	El requisito origen está relacionado con el requisito destino de alguna forma.
Requirement priority (prioridad_requisito)	High	
	Medium	
	Low	
Criticity (criticidad_requisito)	High	
	Medium	
	Low	
Requirement risk (riesgo_requisito)	High	
	Medium	
	Low	
Requirement state (estado_requisito)	Still to be	

Vocabulario	Término	Descripción
	Defined	
	Pending review	
	Descarded	
	Approved	
	Modeling analysis	
	Modeling design	
	Implemented	
	Verified	
	Pending discussion	
Requirement compliance (cumplimiento_requisito)	Required	
	Recommendable	
	Optional	
Verification method (metodo_verif_requisito)	Inspection	
	Analysis	
	Demostration	
	Test	

Tabla 27 - Resumen términos de vocabularios

5.4.2 Tipos de contenido

Los tipos de contenido en Drupal son colecciones de tipos de campos que están relacionados entre ellos por un contexto de información.

Es la forma que tiene Drupal para gestionar el contenido como CMS.

Contamos con una amplia variedad de tipos de datos, desde enteros hasta tipos imagen, ampliable con módulos que nos permite poder gestionar contenido de cualquier tipo.

Los tipos de contenido se manejan en Drupal como objetos node, también llamados nodos.

Un nodo tiene un identificador numérico único asignado por Drupal mediante el cual se puede acceder al nodo y realizar operaciones con él.

En nuestro módulo vamos a crear varios tipos de contenido según el dominio de PANGEA. Estos tipos de contenido se crearán con la función pantalasa_cms_crear_content_types(). Para cada tipo de contenido a crear lo implementaremos en un fichero bajo el directorio /content_types y la función pantalasa_cms_crear_content_types(), al cargar el fichero, se instalará el tipo de contenido.

La forma de crear tipos de contenido en Drupal es a través del módulo Node incluido en el core de Drupal. Para crear un nuevo tipo de contenido disponemos de la función node_type_save() que recibe como parámetro un array con unos campos determinados que principalmente son: nombre máquina del tipo de contenido, nombre descriptivo, base del tipo, descripción textual, información de ayuda y un título.

Presentamos una forma de creación genérica de tipos de contenido:

```
$content_type = array(
      'type' => 'machine_name',
      'name' => 'Type name',
      'base' => 'base_type',
      'description' => 'Descripción textual',
      'help' => 'Información de ayuda',
      'title_label' => 'Etiqueta del título del tipo',
);
node_type_save($content_type);
```

Podemos encontrar en la referencia de la función [29] más información acerca de los índices del array y para qué sirven. Únicamente vamos a señalar el índice "title_label". Cada tipo de contenido tiene un campo obligatorio, "title". Todos los tipos de contenido tienen este campo y cuando se crea un nuevo tipo se crea el campo "title" automáticamente y con "title_label" añadimos la etiqueta que mostrará Drupal a la hora de crear un nuevo tipo de contenido, en el campo "title", aunque nosotros no usaremos los formularios estándar que proporciona Drupal para crear tipos de contenido sino que crearemos nuestros propios formularios para mejorar la usabilidad usando la API Form de Drupal.

Una vez creado el tipo de contenido podemos añadir nuevos campos o más bien, instancias de campos. En Drupal podemos definir campos que pueden ser usados en varios tipos de contenido, es decir, lo que tenemos un tipo de contenido no es un campo sino una instancia de un campo. Así pues, se puede definir, por ejemplo, un campo "Resumen" de tipo "text" y tener una instancia "Resumen" en un tipo "Artículo" y otra instancia "Resumen" en un tipo "Libro".

Para crear un campo disponemos del módulo Field disponible en el core de Drupal. Ese módulo nos proporciona la función field_create_field() que recibe como parámetro un array con determinados valores entre los que destacamos los siguientes: cardinalidad, nombre del campo, etiqueta descriptiva, tipo de campo y preferencias.

En la documentación de la función [30] podemos obtener información detallada de cada valor del array. Aquí sólo describiremos brevemente cada valor. El índice "cardinality" indica cuantos campos de ese tipo habrá en un tipo de contenido. Normalmente este valor es 1 pero como vamos a usar el módulo "References" podemos tener varios campos referenciando a un contenido. "field_name" establece el nombre máquina del campo, "label" establece la etiqueta descriptiva del mismo, "type" indica qué tipo de campo es, texto, imagen, número, etc. Por último tenemos el índice "settings" que es un array que aplica restricciones al campo.

Podemos ver en código como se crea un campo:

```
$field = array(
      'cardinality' => 1,
      'field_name' => 'machine_name',
      'label' => 'Etiqueta descriptiva',
      'type' => 'tipo_de_campo',
      'settings' => array(
```

```
            ...
        ),
    );
    field_create_field($field);
```

Si nos fijamos, no hemos indicado en ninguna parte el tipo de contenido que tiene ese campo. Esto es debido a que es un campo genérico y no una instancia. Una vez hemos creado el campo podemos crear una instancia concreta para un tipo de contenido.

Para crear la instancia de un campo disponemos de la función field_create_instance() del módulo Field, que recibe como parámetro un array con determinados valores. Principalmente los valores son el nombre del campo, la entidad que tendrá la instancia del campo, el tipo de contenido que contendrá el campo, una etiqueta descriptiva, una descripción textual y un componente.

El nombre del campo es el nombre máquina del campo genérico, la entidad se refiere al tipo de entidad que va a contener la instancia: si es un tipo de contenido será “node”, si es un usuarios será “user”, etc. El tipo de contenido indica el nombre máquina del tipo de contenido que tendrá el campo y por último, el componente se refiere al tipo de componente concreto que representa el campo, por ejemplo, si en la definición del campo genérico hemos definido que es de tipo “text” como componente diremos que usaremos un “textfield” aunque puede haber otros componentes para el tipo “text”.

Vamos a ver en código cómo se crea la instancia de un campo:

```
    $instance = array(
        ‘field_name’ => ‘machine_name_field’,
        ‘entity_type => ‘entity (node, user, ...)’
        ‘label’ => ‘etiqueta descriptiva’,
        ‘description’ => ‘Descripción textual’,
        ‘bundle’ => ‘content_type_machine_name’,
        ‘widget’ => array(
            ‘type’ => ‘tipo_de_componente’,
        ),
    );
    field_create_instance($instance);
```

Esta es la forma genérica que usaremos para crear los tipos de contenido y sus campos. A continuación vamos a detallar los tipos de contenido a crear con los campos que tienen:

5.4.2.1 Catálogo

Este tipo de contenido representa un catálogo de requisitos. Un catálogo de requisitos es una colección de requisitos y documentos de requisitos, sin embargo, en nuestra implementación solamente vamos a referenciar a los requisitos que no están organizados en documentos. No vamos a referencias a los documentos de un catálogo debido a que a la hora de trabajar con la herramienta es más cómo hacerlo desde documentos y requisitos y las operaciones que se van a implementar hacen, principalmente, uso de documentos y componentes de documentos (requisitos y secciones). Por tanto, es más fácil recuperar un catálogo desde un documento

que cada vez que queramos realizar alguna operación sobre un documento tener que buscar en todos los catálogos dicho documento.

A la hora de trabajar con permisos y hacer comprobaciones de seguridad también es más eficiente hacer la comprobación desde un documento ya que podemos comprobar si un usuario tiene acceso o no a un catálogo o proyecto directamente recuperando el catálogo o proyecto.

Vamos a describir en la los valores asignados al tipo catálogo:

Atributo	Valor
type	catalogo_type
name	Catalog
base	node_content
description	Reusable requirements catalogue.
help	A catalog is a reusable collection of related requirements for a profile.
custom	TRUE
hash_title	TRUE
title_label	Catalog name

Añadiremos un campo de tipo “body” para poder permitir al usuario incluir una descripción textual del catálogo. El módulo “node” permite añadir ese campo específico con la función node_add_body_field() que recibe como parámetros el objeto tipo de contenido que contendrá el campo y una cadena de texto con la etiqueta del campo.

5.4.2.1.1 Requisitos

Para representar la colección de requisitos sin organizar en documentos vamos a crear un campo propio llamado cat_reqs_field de un número ilimitado de referencias a nodos de tipo requisito. Para indicar que es una referencia a un nodo tenemos el tipo “node_reference” que nos proporciona el módulo “Node References”. La forma de declarar este campo se hace con los valores que se muestran en la **¡Error! No se encuentra el origen de la referencia.**:

Atributo	Valor
cardinality	FIELD_CARDINALITY_UNLIMITED
field_name	cat_reqs_field
label	Requirements
type	node_reference
settings	Array con índice “referenceable_types” cuyo valor es un array con índice “requisito_type” con valor “requisito_type”.

Nota: FIELD_CARDINALITY_UNLIMITED es una constante de Drupal que indica que habrá un número ilimitado de campos.

Nota: El tipo requisito_type será definido posteriormente.

Ahora definimos la instancia del campo. Será una instancia del campo creado, “cat_reqs_field” sobre el tipo de contenido “catalogo_type”. Al ser un tipo de contenido representado en Drupal como un nodo, debemos definir el atributo “entity_type” como un nodo (“node”).

El tipo de componente, en principio es indiferende debido a que crearemos un formulario propio de creación de catálogos. Lo dejaremos como un "node_reference_autocomplete" que es un campo de texto con predicción.

El número de requisitos puede ser ilimitado aunque podría no tener requisitos como puede ser en el momento de su creación. Por este motivo no será un campo requerido en el momento de creación.

Los valores de la instancia son los siguientes:

Atributo	Valor
field_name	cat_reqs_field
entity_type	node
bundle	catalogo_type
label	Requirements
description	Requirements not rated
widget	Array con índice "type" con valor "node_reference_autocomplete".
required	0

5.4.2.2 Proyecto

Un proyecto es una colección de requisitos relacionados por un proyecto de desarrollo de algún tipo de producto. Como tipo de contenido a nivel Drupal, es muy similar al tipo de contenido catálogo. Ambos tipos tiene un campo título, un campo descripción y un campo requisitos para referenciar a los nodos de tipo requisito que representan los requisitos sin organizar en documentos que pueda tener un proyecto.

Las razones de por qué sí referenciar a requisitos no organizados en documentos y no referenciar a los documentos son las mismas que las que se han dado para un catálogo. Es más, un documento tienen campo que bien puede referenciar a un catálogo o a un proyecto. Más adelante hablaremos más en profundidad del tipo de contenido que representa a un Documento.

Vamos a describir los valores asignados al tipo de contenido "Proyecto":

Atributo	Valor
type	proyecto_type
name	Project
base	node_content
description	Allow you to manage software project requirements.
help	Instance of a software project.
custom	TRUE
hash_title	TRUE
title_label	Project name

Asignamos también el campo "body" que servirá para que el usuario introduzca una descripción del proyecto.

5.4.2.2.1 Requisitos

Ahora vamos a crear el campo que representa los requisitos. Si bien es verdad que podríamos aprovechar el campo creado en el tipo de contenido catálogo y crear una instancia de ese campo pero hemos decidido crear un nuevo campo para no ligar las representaciones de proyecto y catálogo y permitir que ambas evolucionen de forma separada en futuras ampliaciones de la herramienta y mejoras del módulo PANTALASA_CMS.

El campo creado tiene lo valores a continuación se muestran:

Atributo	Valor
cardinality	FIELD_CARDINALITY_UNLIMITED
field_name	proy_reqs_field
label	Requirements
type	node_reference
settings	Array con índice “referenceable_types” cuyo valor es un array con índice “requisito_type” con valor “requisito_type”.

Ahora crearemos la instancia del campo. Lo más importante al definir la instancia es el tipo de contenido al que se asocia y de qué tipo es la instancia ya que otros atributos como la obligatoriedad, etiqueta o componente no son de vital importancia debido a que crearemos nuestro propio formulario de creación y edición para el tipo de contenido “Proyecto”. No obstante, asignaremos unos valores a estos campos a fin de facilitar tareas de administración por parte del usuario administrador de Drupal. Los valores son muy similares a los establecidos en la instancia del campo para representar requisitos en el tipo de contenido catálogo. A continuación se muestran:

Atributo	Valor
field_name	proy_reqs_field
entity_type	node
bundle	proyecto_type
label	Requirements
description	Requirements not rated
widget	Array con índice “type” con valor “node_reference_autocomplete”.
required	0

Creamos también un campo textual para representar las keywords de un proyecto y poder facilitar las búsquedas:

Atributo	Valor
field_name	keywords_proyecto_field
label	Keywords
type	text

La instancia de este campo es la siguiente:

Atributo	Valor
field_name	keywords_proyecto_field

Atributo	Valor
entity_type	node
bundle	proyecto_type
label	Keywords
description	Keywords
widget	Array con índice “type” con valor “textfield”.
required	0

5.4.2.3 Documento

El tipo de contenido “Documento” representa a un documento de requisitos. Los documentos contienen requisitos y secciones de requisitos.

De la misma forma que para un proyecto o un catálogo, un documento puede tener requisitos sin clasificar en secciones de forma que añadiremos un campo similar a los que ya hemos creado para referenciar a los requisitos de un documento. También referencia al catálogo o proyecto al que pertenece, tiene un estado y es de un tipo. También tendrá un campo “body” para que el usuario introduzca una descripción textual del documento.

Los valores para crear el tipo de contenido “Documento” son los siguientes:

Atributo	Valor
Type	documento_type
name	Document
base	node_content
description	Requirements document
help	Contains requirements and requirements sections.
custom	TRUE
hash_title	TRUE
title_label	Document title

Nota: no usaremos el formulario de creación que proporciona Drupal para crear nuevos documento sino que crearemos un formulario personalizado.

5.4.2.3.1 Catálogo/Proyecto

Como hemos mencionado anteriormente, un documento referencia al catálogo o proyecto al que pertenece a modo de facilitar la comprobación de permisos de los usuarios para ejecutar acciones sobre un documento y sus componentes. Por este motivo hemos decidido crear un campo para dicha referencia.

El campo tiene los siguientes valores:

Atributo	Valor
field_name	cp_documento
label	Catalog/Project
type	node_reference
settings	Array con índice “referenceable_types” cuyo valor es un array con dos índices: “catalogo_type” con valor “catalogo_type” y “proyecto_type” con valor “proyecto_type”.

El campo es instanciado con los siguientes valores:

Atributo	Valor
field_name	cp_documento
entity_type	node
bundle	documento_type
label	Catalog/Project
widget	Array con índice “type” con valor “options_select”.

5.4.2.3.2 Requisitos

Para representar los requisitos que tiene un documento que no están organizados en secciones vamos a crear un campo muy similar a los ya creados para catálogo y proyecto. El campo tendrá los siguientes valores:

Atributo	Valor
cardinality	FIELD_CARDINALITY_UNLIMITED
field_name	doc_reqs_field
label	Requirements
type	node_reference
settings	Array con índice “referenceable_types” cuyo valor es un array con índice: “requisito_type” con valor “requisito_type”.

La instancia del campo tiene los valores siguientes:

Atributo	Valor
field_name	doc_reqs_field
entity_type	node
bundle	documento_type
label	Requirements
description	Requirements not rated
widget	Array con índice “type” con valor “node_reference_autocomplete”.
required	0

Los siguientes dos campos representan el tipo y estado de un documento. Podemos decir que los valores de estos campos son finitos y es por eso por lo que los hemos representado como vocabularios. En la sección de vocabularios de detallan todos los vocabularios y sus términos. Para el campo tipo de documento vamos tenemos el vocabulario ”Document type” y para el estado tenemos “Document state”.

Para crear un campo que sea una referencia a un término de un vocabulario disponemos del tipo “taxonomy_term_reference” que nos facilita mucho las cosas. Como ajustes del campo tenemos que decirle a qué vocabularios puede referenciar sus términos.

5.4.2.3.3 Tipo de documento

Así pues, para el campo tipo de documento, establecemos los siguientes valores:

Atributo	Valor
field_name	tipo_documento
type	taxonomy_term_reference
settings	Array con índice “allowed_values” cuyo valor es un array que contiene otro array con dos índices: “vocabulary” con valor “tipo_documento” y “parent” con valor 0.

La instancia de este campo se crea con los valores siguientes:

Atributo	Valor
field_name	tipo_documento
entity_type	node
bundle	documento_type
Label	Document type
widget	Array con índice “type” con valor “options_select”.
required	true

5.4.2.3.4 Estado de documento

Para el campo “estado” la única diferencia respecto al campo anterior es el vocabulario que referencia:

Atributo	Valor
field_name	estado_documento
type	taxonomy_term_reference
settings	Array con índice “allowed_values” cuyo valor es un array que contiene otro array con dos índices: “vocabulary” con valor “estado_documento” y “parent” con valor 0.

Y la instancia:

Atributo	Valor
field_name	estado_documento
entity_type	node
bundle	documento_type
Label	Document state
widget	Array con índice “type” con valor “options_select”.
required	true

5.4.2.4 Sección

Los documentos de requisitos se componen de requisitos y estos se pueden organizar en secciones. A su vez, las secciones se pueden dividir en subsecciones que también contienen requisitos.

En este caso no necesitamos un campo para representar una colección de requisitos organizados en una sección sino que esa relación se establecerá a nivel de requisito. También añadiremos un campo donde introducir el número de sección, un campo de referencia al documento al que pertenece la sección de forma obligatoria y un campo de referencia a una

sección donde está contenida. Este último campo es opcional. Añadiremos también un campo "body" para que el usuario introduzca una descripción textual de la sección de forma opcional.

Los atributos que tiene este tipo de contenido son los siguientes:

Atributo	Valor
Type	seccion_type
name	Section
base	node_content
description	Section of a requirements document
help	Section of a requirements document.
custom	TRUE
hash_title	TRUE
title_label	Section name

Los valores de los campos que vamos a crear serán introducidos a través de un formulario propio. Si bien habrá que establecer un valor determinado para tareas administrativas llevadas a cabo por el administrador de Drupal.

5.4.2.4.1 Número de sección

El campo que define el número de sección será un simple campo de texto definido de la siguiente forma:

Atributo	Valor
field_name	numero_seccion
label	Section number
type	text

Su instancia se establece de la siguiente forma:

Atributo	Valor
field_name	numero_seccion
entity_type	node
bundle	seccion_type
Label	Section number
widget	Array con índice "type" con valor "textfieldt".
required	true

5.4.2.4.2 Documento

El campo que define la relación con el documento al que pertenece tendrá los siguientes atributos:

Atributo	Valor
field_name	documento_seccion
label	Document
type	node_reference
settings	Array con índice "referenceable_types" cuyo valor es otro array con un

	índice “documento_type” y valor “documento_type”.

La instancia de este campo tendrá los siguientes atributos:

Atributo	Valor
field_name	documento_seccion
entity_type	node
label	Document
type	node_reference
widget	Array con índice “type” cuyo valor es “options_select”.
bundle	seccion_type
required	true

5.4.2.4.3 Sección padre

Por último vamos a definir el campo usado para crear la relación con la sección padre:

Atributo	Valor
field_name	field_father_section
label	Parent section
type	node_reference
settings	Array con índice “referenceable_types” cuyo valor es otro array con un índice “seccion_type” y valor “seccion_type”.

La instancia de este campo tiene los siguientes atributos:

Atributo	Valor
field_name	field_father_section
entity_type	node
label	Parent section
type	node_reference
widget	Array con índice “type” cuyo valor es “options_select”.
bundle	seccion_type

5.4.2.5 Parámetro

Con este tipo de contenido vamos a representar los tipos de parámetros posibles para los requisitos parametrizables. Su representación es bastante simple ya que solamente se compone de un nombre, una descripción y un nombre clave.

El nombre clave es el que se usará para crear los posibles valores e instancias. Este es el único campo que vamos adicional que vamos a crear ya que para la descripción podemos usar el campo “body” y para el nombre usar el campo “title”.

Los valores de los atributos para crear el tipo de contenido son los siguientes:

Atributo	Valor
Type	parametro_type
name	Param

Atributo	Valor
base	node_content
description	Param type
help	A param type for parametrizable requirements.
custom	TRUE
hash_title	TRUE
title_label	Type name

5.4.2.5.1 Clave

El campo que representa la clave es un campo de texto:

Atributo	Valor
field_name	clave_parametro
label	Key name
type	text

Y su instancia:

Atributo	Valor
field_name	clave_parametro
entity_type	node
label	Key name
widget	Array con índice “type” cuyo valor es “textfield”.
bundle	parametro_type
required	true

5.4.2.6 Valor de parámetro

Los valores de parámetro son posibles valores para un tipo de parámetro. Este tipo de contenido representa cada uno de esos valores posibles para un tipo de parámetro.

Su represetnación también es bastante simple ya que se trata de un nombre y una descripción textual que sirva de ayuda al usuario. Sólo falta establecer de alguna forma a qué tipo de parámetro está asignado. Esto se hace creando un campo que sea una referencia al tipo de contenido que representa el tipo de parámetro.

Los valores para crear este tipo de contenido son:

Atributo	Valor
Type	valor_parametro_type
name	Param value
base	node_content
description	Possible value for a param type.
help	Value for a param type
custom	TRUE
hash_title	TRUE
title_label	Value

5.4.2.6.1 Tipo de parámetro

El campo que va a representar la relación con el tipo de contenido tiene los siguientes valores:

Atributo	Valor
field_name	parametro_field
label	Param type
type	node_reference
settings	Como valor tiene array con índice "referenceable_types" y cuyo valor es otro array con un índice "parametro_type" y valor "parametro_type".

Los valores de la instancia se muestran en esta tabla:

Atributo	Valor
field_name	parametro_field
entity_type	node
label	Param type
type	node_reference
widget	Array con índice "type" cuyo valor es "options_select".
bundle	valor_parametro_type
required	true

5.4.2.7 Instancia de parámetro

La instancia de parámetro indica para un requisito, qué valor tiene un tipo de parámetro concreto.

Las instancias de este tipo de dato se crearán en el momento de instanciar una serie de parámetros de un requisito.

El campo "title" es obligatorio en Drupal de forma que aunque no lo vamos a usar ya que no tiene sentido añadir ningún título a una instancia no lo podemos eliminar. Lo dejaremos siempre vacío. De igual modo tampoco es útil un campo "body" de manera que no se añadirá.

Vamos a añadir tres nuevos campos para representar las relaciones que va a tener la instancia de un parámetro: el requisito al que pertenece, el tipo de parámetro y el valor asignado.

La creación de este tipo de contenido tiene los siguientes valores:

Atributo	Valor
Type	instancia_parametro_type
name	Param instance
base	node_content
description	Param instance in a requirement.
help	Param instance in a requirement.
custom	TRUE
hash_title	TRUE
title_label	Instance name

5.4.2.7.1 Valor de parámetro

El campo valor del parámetro tiene las siguientes propiedades:

Atributo	Valor
field_name	instancia_valor_field
label	Value
type	node_reference
settings	Como valor tiene array con índice “referenceable_types” y cuyo valor es otro array con un índice “valor_parametro_type” y valor “valor_parametro_type”.

Y su instancia:

Atributo	Valor
field_name	instancia_valor_field
entity_type	node
label	Value
type	node_reference
widget	Array con índice “type” cuyo valor es “node_reference_autocomplete”.
bundle	instancia_parametro_type
required	True

5.4.2.7.2 Tipo de parámetro

El campo que representa la relación con el tipo de parámetro concreto tiene los siguientes valores:

Atributo	Valor
field_name	instancia_param_field
label	Param
type	node_reference
settings	Como valor tiene array con índice “referenceable_types” y cuyo valor es otro array con un índice “parametro_type” y valor “parametro_type”.

Y su instancia:

Atributo	Valor
field_name	instancia_param_field
entity_type	node
label	Param
type	node_reference
widget	Array con índice “type” cuyo valor es “node_reference_autocomplete”.
bundle	instancia_parametro_type
required	True

5.4.2.7.3 Requisito

Finalmente, el campo que referencia al requisito será una referencia de nodo de la siguiente forma:

Atributo	Valor
field_name	instancia_req_field
label	Requirement
type	node_reference
settings	Como valor tiene array con índice “referenceable_types” y cuyo valor es otro array con un índice “requisito_type” y valor “requisito_type”.

Y su instancia:

Atributo	Valor
field_name	instancia_req_field
entity_type	node
label	Requirement
type	node_reference
widget	Array con índice “type” cuyo valor es “node_reference_autocomplete”.
bundle	instancia_parametro_type
required	True

5.4.2.8 Traza

Con este tipo de contenido vamos a representar las relaciones existentes entre requisitos desde el punto de vista de un requisito origen, es decir, será un requisito quien tenga un conjunto de trazas a otros requisitos.

En cuanto a campos necesarios, pese a que Drupal crea un campo título obligatorio no tiene mucho sentido este campo pero lo usaremos a modo de pequeña descripción de la traza.

Necesitamos representar también el requisito origen y el requisito destino. Ambos campos serán una referencia a un nodo de tipo requisito. También hemos de representar el tipo de traza por lo que necesitamos un campo de tipo referencia de término al vocabulario que tiene los términos de tipos de trazas, “tipo_traza”.

El tipo de contenido tiene los siguientes atributos:

Atributo	Valor
Type	traza_type
name	Requirement trace
base	node_content
description	Relationship with another requirement.
help	With the traces we can define the relationship you have a requirement to another.
custom	TRUE
hash_title	TRUE
title_label	Trace description

5.4.2.8.1 Requisito origen

El campo requisito origen tiene la siguiente forma:

Atributo	Valor
field_name	traza_requisito_origen
label	Requirement
type	node_reference
settings	Como valor tiene array con índice “referenceable_types” y cuyo valor es otro array con un índice “requisito_type” y valor “requisito_type”.

Y su instacia:

Atributo	Valor
field_name	traza_requisito_origen
entity_type	node
label	Requirement
type	node_reference
widget	Array con índice “type” cuyo valor es “options_select”.
Bundle	traza_type
required	True

5.4.2.8.2 Requisito destino

El campo requisito destino tiene la siguiente forma:

Atributo	Valor
field_name	traza_requisito_destino
label	Requirement
type	node_reference
settings	Como valor tiene array con índice “referenceable_types” y cuyo valor es otro array con un índice “requisito_type” y valor “requisito_type”.

Y su instacia:

Atributo	Valor
field_name	traza_requisito_destino
entity_type	node
label	Requirement
type	node_reference
widget	Array con índice “type” cuyo valor es “options_select”.
Bundle	traza_type
required	True

5.4.2.8.3 Tipo de traza

Por último, el campo que representa el tipo de traza:

Atributo	Valor
field_name	tipo_traza
type	taxonomy_term_reference

settings	Como valor tiene array con índice “allowed_values” y cuyo valor es otro array que contiene otro array con un índice “vocabulary” y valor “tipo_traza” y otro índice “parent” con valor 0.
	`'settings' => array (`
	`'allowed_values' => array (`
	`array (`
	`'vocabulary' => 'tipo_traza',`
	`'parent' => 0,`
	`),`
	`),`
	`),`

Y su instancia:

Atributo	Valor
field_name	tipo_traza
entity_type	node
label	Trace type
widget	Array con índice “type” cuyo valor es “options_select”.
Bundle	traza_type
required	True

5.4.2.9 Hilo de discusión

Un hilo de discusión es un tipo de contenido relacionado con un requisito que permite recibir comentarios.

Aprovecharemos el sistema de comentarios que proporciona Drupal para permitir añadir comentarios a instancias de este tipo de contenido.

Los campos que tendrá este tipo de contenido serán una descripción, para la cual aprovecharemos el campo “body” que nos proporciona Drupal, una referencia al requisito al que se alude en el hilo y un estado de hilo de discusión, que será una referencia de término al vocabulario que tiene los términos posibles de estado de un hilo.

El campo “title” lo usaremos para introducir un nombre descriptivo resumen del hilo.

Los valores para crear este tipo de contenido son:

Atributo	Valor
type	hilo_type
name	Discussion thread
base	node_content
description	Requirement discussion thread.
help	Through a discussion thread can discuss aspects of a requirement.

Atributo	Valor
custom	TRUE
hash_title	TRUE
title_label	Thread title

5.4.2.9.1 Requisito

El campo que referencia al requisito al que se hace alusión en el hilo tiene la siguiente forma:

Atributo	Valor
field_name	requisito_hilo
label	Requirement
type	node_reference
settings	Como valor tiene array con índice “referenceable_types” y cuyo valor es otro array con un índice “requisito_type” y valor “requisito_type”.

Y su instancia:

Atributo	Valor
field_name	requisito_hilo
entity_type	node
label	Requirement
type	node_reference
widget	Array con índice “type” cuyo valor es “options_select”.
Bundle	traza_type
required	True

5.4.2.9.2 Estado del hilo

Por último, el campo que representa el estado del hilo:

Atributo	Valor
field_name	estado_hilo_hilo
type	taxonomy_term_reference

settings	Como valor tiene array con índice “allowed_values” y cuyo valor es otro array que contiene otro array con un índice “vocabulary” y valor “estado_hilo” y otro índice “parent” con valor 0.
	`'settings' => array (`
	`'allowed_values' => array (`
	`array (`
	`'vocabulary' => ' estado_hilo',`
	`'parent' => 0,`
	`),`
	`),`
	`),`

Y su instancia:

Atributo	Valor
field_name	estado_hilo_hilo
entity_type	node
label	Thread state
widget	Array con índice “type” cuyo valor es “options_select”.
bundle	hilo_type
required	True

5.4.2.10 Grupo

Con este tipo de contenido vamos a representar a los grupos de trabajo en un proyecto.

Cada grupo de trabajo tendrá un conjunto de usuarios miembro, un usuario representante y pertenecerá a un proyecto. Estos son los tres campos que necesitamos crear además del campo “title” que aprovecharemos para asignar un nombre identificativo al grupo y añadiremos también un campo “body” para incluir una descripción textual del grupo.

El tipo de contenido grupo tiene los siguientes valores:

Atributo	Valor
type	grupo_type
name	Group
base	node_content
description	Workgroup.
help	An user set that work in a project
custom	TRUE
hash_title	TRUE
title_label	Group name

5.4.2.10.1 Miembros

El campo que representa los usuarios miembros tiene los siguientes valores:

Atributo	Valor
cardinality	FIELD_CARDINALITY_UNLIMITED
field_name	miembros_grupo_field
label	Members
type	user_reference
settings	Como valor tiene array con índice “referenceable_roles” y cuyo valor es otro array vacío.

Nota: El control de los roles referenciados lo haremos de forma manual en el formulario que usaremos para crear los grupos para evitar inconsistencias en la instalación del módulo ya que no es posible referenciar en el momento de creación del tipo de contenido al rol correspondiente.

La instancia de este campo tiene la siguiente forma:

Atributo	Valor
field_name	miembros_grupo_field
entity_type	node
label	Members.
description	Members.
widget	Array con índice “type” cuyo valor es “user_reference_autocomplete”.
bundle	grupo_type
required	0

5.4.2.10.2 Representante de equipo

El campo para referenciar al representante de equipo tiene una forma muy similar salvo que sólo se permite referenciar a un representante. Los valores son los siguientes:

Atributo	Valor
cardinality	1
field_name	repr_grupo_field
label	Agent
type	user_reference
settings	Como valor tiene array con índice “referenceable_roles” y cuyo valor es otro array vacío.

Y su instancia:

Atributo	Valor
field_name	repr_grupo_field
entity_type	node
label	Agent.
description	Workgroup agent.
widget	Array con índice “type” cuyo valor es “user_reference_autocomplete”.
bundle	grupo_type
required	TRUE

5.4.2.10.3 Proyecto

Por último, el campo que referencia al proyecto al que pertenece el grupo:

Atributo	Valor
cardinality	1
field_name	grupo_proyecto_field
label	Project
type	node_reference
settings	Como valor tiene array con índice “referenceable_types” y cuyo valor es otro array cón índice “proyecto_type” y valor “proyecto_type”.

Y su instancia:

Atributo	Valor
field_name	grupo_proyecto_field
entity_type	node
label	Project.
description	Project to which the group.
widget	Array con índice “type” cuyo valor es “node_reference_autocomplete”.
bundle	grupo_type
required	0

5.4.2.11 Requisito

Este tipo de contenido representa a un requisito. Tiene una gran cantidad de campos debido a la cantidad de información asociada que necesita. Hay tres tipos de campos, referencias a un vocabulario, referencias a un usuario, referencias a un nodo y campos textuales.

Todos los campos son ya conocidos y hemos explicado cómo sería su creación. Nos limitaremos en este punto a citar los valores de creación e instanciación campo a campo.

Añadir que usaremos el campo “title” como un nombre del requisito sin más trascendencia que la de identificarlo de forma rápida por un nombre. Usaremos el campo “body” como texto del requisito.

5.4.2.11.1 Identificador de requisito

Campo:

Atributo	Valor
field_name	id_requisito_field
label	Requirement identifier
type	text

Instancia:

Atributo	Valor
field_name	id_requisito_field
entity_type	node
label	Requirement identifier.
description	Requirement identifier.
widget	Array con índice “type” cuyo valor es “textfield”.
bundle	requisito_type

5.4.2.11.2 Sección del requisito

Campo:

Atributo	Valor
field_name	seccion_requisito

Atributo	Valor
label	Section
type	node_reference
settings	Como valor tiene array con índice “referenceable_types” y cuyo valor es otro array cón índice “seccion_type” y valor “seccion_type”.

Instancia:

Atributo	Valor
field_name	seccion_requisito
entity_type	node
label	Section.
description	Section requirement.
widget	Array con índice “type” cuyo valor es “options_select”.
bundle	requisito_type

5.4.2.11.3 Riesgo

Campo:

Atributo	Valor
field_name	riesgo_requisito_field
label	Requirement risk
type	taxonomy_term_reference

settings	Como valor tiene array con índice “allowed_values” y cuyo valor es otro array que contiene otro array con un índice “vocabulary” y valor “riesgo_requisito” y otro índice “parent” con valor 0.
	
	`'settings' => array (`
	`'allowed_values' => array (`
	`array (`
	`'vocabulary' => 'riesgo_requisito',`
	`'parent' => 0,`
	`),`
	`),`
	`),`

Instancia:

Atributo	Valor
field_name	riesgo_requisito_field
entity_type	node
label	Requirement risk.
description	Requirement risk.
widget	Array con índice “type” cuyo valor es “options_select”.
bundle	requisito_type
required	TRUE

5.4.2.11.4 Criticidad

Campo:

Atributo	Valor
field_name	criticidad_requisito_field
label	Criticity
type	taxonomy_term_reference

settings	Como valor tiene array con índice “allowed_values” y cuyo valor es otro array que contiene otro array con un índice “vocabulary” y valor “criticidad_requisito” y otro índice “parent” con valor 0.
	
	`'settings' => array (`
	`'allowed_values' => array (`
	`array (`
	`'vocabulary' => 'criticidad_requisito',`
	`'parent' => 0,`
	`),`
	`),`
	`),`

Instancia:

Atributo	Valor
field_name	criticidad_requisito_field
entity_type	node
label	Criticality of the requirement.
description	Criticality of the requirement.
widget	Array con índice “type” cuyo valor es “options_select”.
bundle	requisito_type
required	TRUE

5.4.2.11.5 Estado

Campo:

Atributo	Valor
field_name	estado_requisito_field
label	State
type	taxonomy_term_reference

settings	Como valor tiene array con índice “allowed_values” y cuyo valor es otro array que contiene otro array con un índice “vocabulary” y valor “estado_requisito” y otro índice “parent” con valor 0.
	
	`'settings' => array (`
	`'allowed_values' => array (`
	`array (`
	`'vocabulary' => 'estado_requisito',`
	`'parent' => 0,`
	`),`

Atributo	Valor
	),),

Instancia:

Atributo	Valor
field_name	estado_requisito_field
entity_type	node
label	State requirement
description	State requirement
widget	Array con índice “type” cuyo valor es “options_select”.
bundle	requisito_type
required	TRUE

5.4.2.11.6 Prioridad

Campo:

Atributo	Valor
field_name	prioridad_requisito_field
label	Priority
type	taxonomy_term_reference
settings	Como valor tiene array con índice “allowed_values” y cuyo valor es otro array que contiene otro array con un índice “vocabulary” y valor “prioridad_requisito” y otro índice “parent” con valor 0. 'settings' => array ('allowed_values' => array (array ('vocabulary' => 'prioridad_requisito', 'parent' => 0,),),),

Instancia:

Atributo	Valor
field_name	prioridad_requisito_field
entity_type	node
label	Requirement priority
description	Requirement priority
widget	Array con índice “type” cuyo valor es “options_select”.
bundle	requisito_type
required	TRUE

5.4.2.11.7 Cumplimiento

Campo:

Atributo	Valor
field_name	cumplimiento_requisito_field
label	Requirement compliance
type	taxonomy_term_reference

settings	Como valor tiene array con índice “allowed_values” y cuyo valor es otro array que contiene otro array con un índice “vocabulary” y valor “cumplimiento_requisito” y otro índice “parent” con valor 0.
	
	`'settings' => array (`
	`'allowed_values' => array (`
	`array (`
	`'vocabulary' => 'cumplimiento_requisito',`
	`'parent' => 0,`
	`),`
	`),`
	`),`

Instancia:

Atributo	Valor
field_name	cumplimiento_requisito_field
entity_type	node
label	Requirement compliance
description	Requirement compliance
widget	Array con índice “type” cuyo valor es “options_select”.
bundle	requisito_type
required	TRUE

5.4.2.11.8 Método de verificacación

Campo:

Atributo	Valor
field_name	verif_requisito_field
label	Verification method
type	taxonomy_term_reference

settings	Como valor tiene array con índice “allowed_values” y cuyo valor es otro array que contiene otro array con un índice “vocabulary” y valor “metodo_verif_requisito” y otro índice “parent” con valor 0.
	
	`'settings' => array (`
	`'allowed_values' => array (`
	`array (`
	`'vocabulary' => 'metodo_verif_requisito',`
	`'parent' => 0,`

	),),),

Instancia:

Atributo	Valor
field_name	verif_requisito_field
entity_type	node
label	Verification method
description	Verification method
widget	Array con índice “type” cuyo valor es “options_select”.
bundle	requisito_type
required	TRUE

5.4.2.11.9 Motivación

Campo:

Atributo	Valor
field_name	motivacion_requisito_field
label	Motivation
type	text

Instancia:

Atributo	Valor
field_name	motivacion_requisito_field
entity_type	node
label	Motivation
description	Motivation
widget	Array con índice “type” cuyo valor es “textfield”.
bundle	requisito_type

5.4.2.11.10 Fuente

Campo:

Atributo	Valor
field_name	fuente_requisito_field
label	Source
type	text

Instancia:

Atributo	Valor
field_name	fuente_requisito_field
entity_type	node

Atributo	Valor
label	Source
description	Source
widget	Array con índice “type” cuyo valor es “textfield”.
bundle	requisito_type

5.4.2.11.11 Propuesto por

Campo:

Atributo	Valor
cardinality	1
field_name	propuesto_requisito_field
label	Proposed by
type	user_reference
settings	Como valor tiene array con índice “referenceable_roles” y cuyo valor es otro array vacío.

Instancia:

Atributo	Valor
field_name	propuesto_requisito_field
entity_type	node
label	Proposed by
description	Proposed by
widget	Array con índice “type” cuyo valor es “options_select”.
bundle	requisito_type
required	TRUE

5.4.2.11.12 Grupo de trabajo fuente

Campo:

Atributo	Valor
field_name	grupo_fuente_requisito_field
label	Source workgroup
type	node_reference
settings	Como valor tiene array con índice “referenceable_types” y cuyo valor es otro array cón índice “grupo_type” y valor “grupo_type”.

Instancia:

Atributo	Valor
field_name	grupo_fuente_requisito_field
entity_type	node
label	Source workgroup.
description	Source workgroup.
widget	Array con índice “type” cuyo valor es “options_select”.
bundle	requisito_type

5.4.2.11.13 Grupo de trabajo destino

Campo:

Atributo	Valor
field_name	grupo_destino_requisito_field
label	Destination workgroup
type	node_reference
settings	Como valor tiene array con índice “referenceable_types” y cuyo valor es otro array cón índice “grupo_type” y valor “grupo_type”.

Instancia:

Atributo	Valor
field_name	grupo_destino_requisito_field
entity_type	node
label	Destination workgroup.
description	Destination workgroup.
widget	Array con índice “type” cuyo valor es “options_select”.
bundle	requisito_type

5.4.2.11.14 Analista

Campo:

Atributo	Valor
cardinality	1
field_name	analista_fuente_requisito_field
label	Analyst source
type	user_reference
settings	Como valor tiene array con índice “referenceable_roles” y cuyo valor es otro array vacío.

Instancia:

Atributo	Valor
field_name	analista_fuente_requisito_field
entity_type	node
label	Analyst source
description	Analyst source
widget	Array con índice “type” cuyo valor es “options_select”.
bundle	requisito_type
required	TRUE

Creamos también un campo textual para representar las keywords de un requisito y poder facilitar las búsquedas:

Atributo	Valor
field_name	keywords_requisito_field
label	Keywords
type	text

La instancia de este campo es la siguiente:

Atributo	Valor
field_name	keywords_requisito_field
entity_type	node
bundle	requisito_type
label	Keywords
description	Keywords
widget	Array con índice “type” con valor “textfield”.
required	0

5.4.2.12 Resumen de tipos de contenido

En la Tabla 28 se muestra un resumen de los tipos de contenido creados junto con sus campos:

Tipo de Contenido	Nombre de máquina	Campo	Nombre campo
Catálogo	catalogo_type	Requisitos	cat_reqs_field
Proyecto	proyecto_type	Requisitos	proy_reqs_field
Documento	documento_type	Catálogo/Proyecto	cp_documento
		Requisitos	doc_reqs_field
		Tipo de documento	tipo_documento
		Estado de documento	estado_documento
Sección	sección_type	Número	numero_seccion
		Documento	documento_seccion
		Sección padre	field_father_section
Parámetro	parametro_type	Clave	clave_parametro
Valor parámetro	valor_parametro_type	Tipo de parámetro	parametro_field
Traza	traza_type	Requisito	traza_requisito_origen
		Requisito	traza_requisito_destino
		Tipo de traza	tipo_traza
Hilo de discusión	hilo_type	Requisito	requisito_hilo
		Estado del hilo	estado_hilo_hilo
Grupo	grupo_type	Miembros	miembros_grupo_field
		Representante	repr_grupo_field
		Proyecto	grupo_proyecto_field
Requisito	requisito_type	Sección	seccion_requisito
		Identificador	id_requisito_field
		Prioridad	prioridad_requisito_field

Tipo de Contenido	Nombre de máquina	Campo	Nombre campo
		Criticidad	criticidad_requisito_field
		Viabilidad	viabilidad_requisito_field
		Riesgo	riesgo_requisito_field
		Motivación	motivación_field
		Estado actual	estado_requisito_field
		Cumplimiento	cumplimiento_field
		Método de verificación	verif_requisito_field
		Criterios de validación	valid_requisito_field
		Solicitador por	solicitado_requisito_field
		Responsable	responsable_field
		Keywords	kewords_requisito_field

Tabla 28 - Resumen de tipos de contenido y campos

5.4.3 Menú de PANTALASA_CMS

No debemos confundir el menú de PANTALASA_CMS con el menú de navegación de los usuarios. Cuando hablamos del menú de PANTALASA_CMS nos referimos a las opciones de navegación que nos ofrece nuestro módulo. Por ejemplo, podemos definir una opción de navegación que nos lleve a la implementación de un formulario (o cualquier otra funcionalidad implementada) bajo una URL y podemos, mediante un menú de navegación de usuarios, enlazar a dicha URL.

Para definir las opciones de navegación que vamos a permitir en nuestro módulo hemos de implementar el "hook_menu" de Drupal. Cada opción de navegación es un "ítem" en Drupal y un "ítem" es un array con ciertas claves y valores.

El formato general que presenta cada uno de estos arrays "ítem" es el siguiente:

```
$item['url'] = array(
      'title' => 'Título de la página',
      'page callback' => 'funcion_que_implementa',
      'page arguments' => argumentos a pasar en forma de array,
      'access callback' => 'funcion_que_controla_el acceso',
      'access arguments' => argumentos a pasar en forma de
array,
      'type' => MENU_CALLBACK,
);
```

A continuación vamos a listar en la Tabla 29 los ítems de menú creados con una pequeña descripción:

Nota: cuando se abra un formulario la función del callback es "drupal_get_form" donde como argumento se pasa la función que crea el formulario. En "Formulario:" colocaremos el valor SI y en "Implementación" indicaremos la función que se pasa como argumento. Si no es formulario, "Implementación" será el valor del callback.

Operación		
Añadir miembro	**URL**	**pantalasa-cms/add-member/%**
	Formulario:	SI
	Implementación:	pantalasa_cms_add_member_from
	Acceso:	pantalasa_cms_add_member_ac
	Descripción:	Abre un formulario para añadir un miembro a un grupo. Recibe como argumento el NID del grupo.
Cerrar hilo de discusión	**URL**	**pantalasa-cms/close-thread/%/%/%**
	Formulario:	NO
	Implementación:	pantalasa_cms_close_thread
	Acceso:	pantalasa_cms_close_thread_ac
	Descripción:	Cambia el estado a CERRADO de un hilo de discusión. Recibe como argumentos el NID del requisito, el NID del proyecto al que pertenece el requisito y el NID del propio hilo.
Crear catálogo	**URL**	**pantalasa-cms/create-catalog**
	Formulario:	SI
	Implementación:	pantalasa_cms_create_catalog_form
	Acceso:	pantalasa_cms_create_catalog_ac
	Descripción:	Abre un formulario para la creación de un nuevo catálogo.
Crear comentario hilo	**URL**	**pantalasa-cms/add-comment/%/%/%**
	Formulario:	SI
	Implementación:	pantalasa_cms_add_comment_form
	Acceso:	pantalasa_cms_add_coment_ac
	Descripción:	Abre un formulario de creación de comentarios sobre un hilo. Recibe como argumentos el NID del requisito, el NID del hilo y el NID del proyecto.
Crear documento	**URL**	**pantalasa-cms/create-document/%**
	Formulario:	SI
	Implementación:	pantalasa_cms_create_documento_form
	Acceso:	pantalasa_cms_create_documento_ac
	Descripción:	Abre un formulario para la creación de un nuevo documento. Recibe como argumento el NID del catálogo o proyecto que contendrá el documento.
Crear documento base	**URL**	**pantalasa-cms/create-base-version/%**
	Formulario:	NO
	Implementación:	pantalasa_cms_create_base_version
	Acceso:	pantalasa_cms_create_base_version_ac
	Descripción:	Crear una versión base de un documento de requisitos. Recibe como argumento el NID del documento de requisitos a crear la versión base.
Crear grupo	**URL**	**pantalasa-cms/create-group**
	Formulario:	SI
	Implementación:	pantalasa_cms_gest_group_from

Operación		
	Acceso:	pantalasa_cms_create_group_ac
	Descripción:	Abre un formulario de creación de un nuevo grupo.
Crear hilo de discusión	**URL**	**pantalasa-cms/create-discussion-thread/%/%**
	Formulario:	SI
	Implementación:	pantalasa_cms_create_discussion_thread_from
	Acceso:	pantalasa_cms_create_thread_ac
	Descripción:	Abre un formulario de creación de un hilo de discusión. Recibe como argumentos el NID del requisito y el NID del proyecto al que pertenece el requisito.
Crear proyecto	**URL**	**pantalasa-cms/create-project**
	Formulario:	SI
	Implementación:	pantalasa_cms_create_project_form
	Acceso:	pantalasa_cms_create_project_ac
	Descripción:	Abre un formulario para la creación de un nuevo proyecto.
Crear requisito	**URL**	**pantalasa-cms/create-requirement/%**
	Formulario:	SI
	Implementación:	pantalasa_cms_create_requisito_form
	Acceso:	pantalasa_cms_check_permissions_requirement
	Descripción:	Abre un formulario para la creación de un nuevo requisito. Recibe como argumento el NID del tipo de contenido que contendrá el requisito.
Crear sección	**URL**	**pantalasa-cms/add-seccion-documento/%**
	Formulario:	SI
	Implementación:	pantalasa_cms_create_seccion_form
	Acceso:	pantalasa_cms_create_section_ac
	Descripción:	Abre un formulario para la creación de una nueva sección en un documento. Recibe como argumento el NID del documento.
Crear traza	**URL**	**pantalasa-cms/manage-traces/%/%**
	Formulario:	SI
	Implementación:	pantalasa_cms_create_trace_form
	Acceso:	pantalasa_cms_create_traces_ac
	Descripción:	Abre un formulario de creación de trazas. Recibe como argumentos el NID del requisito origen de la traza y el NID del proyecto.
Crear usuario	**URL**	**pantalasa-cms/create-user**
	Formulario:	SI
	Implementación:	pantalasa_cms_create_user_form
	Acceso:	pantalasa_cms_create_edit_user_ac
	Descripción:	Abre un formulario para la creación de usuarios.
Crear valor de parámetro	**URL**	**pantalasa-cms/create-param-value/%**
	Formulario:	SI
	Implementación:	pantalasa_cms_create_param_value_form
	Acceso:	pantalasa_cms_create_param_value_ac

Operación		
	Descripción:	Abre un formulario de creación de un valor de parámetro. Recibe como argumento el NID del tipo de parámetro.
Crear XML	**URL**	**pantalasa-cms/generate-xml/%**
	Formulario:	NO
	Implementación:	pantalasa_cms_generate_xml
	Acceso:	pantalasa_cms_generate_xml_ac
	Descripción:	Genera un documento XML con la información de un documento y su contenido. Recibe como argumento el NID del documento.
Editar catálogo	**URL**	**pantalasa-cms/edit-catalog/%**
	Formulario:	SI
	Implementación:	pantalasa_cms_edit_catalog_form
	Acceso:	pantalasa_cms_edit_catalog_ac
	Descripción:	Abre un formulario para la edición de un catálogo. Recibe como argumento el NID del catálogo a modificar.
Editar documento	**URL**	**pantalasa-cms/edit-document/%/%**
	Formulario:	SI
	Implementación:	pantalasa_cms_create_documento_form
	Acceso:	pantalasa_cms_edit_document_ac
	Descripción:	Abre un formulario para la edición de un documento. Recibe como argumento el NID del catálogo o proyecto que contiene el documento y el NID del documento a modificar.
Editar grupo	**URL**	**pantalasa-cms/edit-group/%**
	Formulario:	SI
	Implementación:	pantalasa_cms_gest_group_from
	Acceso:	pantalasa_cms_edit_group_ac
	Descripción:	Abre un formulario para editar un grupo. Recibe como argumento el NID del grupo a editar.
Editar hilo de discusión	**URL**	**pantalasa-cms/edit-discussion-thread/%/%/%**
	Formulario:	SI
	Implementación:	pantalasa_cms_create_discussion_thread_from
	Acceso:	pantalasa_cms_edit_thread_ac
	Descripción:	Abre un formulario de edición de un hilo de discusión. Recibe como argumentos el NID del requisito, el NID del proyecto al que pertenece el requisito y el NID del propio hilo.
Editar proyecto	**URL**	**pantalasa-cms/edit-project/%**
	Formulario:	SI
	Implementación:	pantalasa_cms_create_project_form
	Acceso:	pantalasa_cms_edit_project_ac
	Descripción:	Abre un formulario para la edición de un proyecto. Como argumento recibe el NID del proyecto.

Operación		
Editar requisito	**URL**	**pantalasa-cms/edit-requirement/%/%**
	Formulario:	SI
	Implementación:	pantalasa_cms_create_requisito_form
	Acceso:	pantalasa_cms_check_permissions_requirement
	Descripción:	Abre un formulario para la edición de un requisito. Recibe como argumento el NID del tipo de contenido que contendrá el requisito y el NID del requisito a modificar.
Editar sección	**URL**	**pantalasa-cms/edit-section/%**
	Formulario:	SI
	Implementación:	pantalasa_cms_edit_section_form
	Acceso:	pantalasa_cms_edit_section_ac
	Descripción:	Abre un formulario para la edición de una sección en un documento. Recibe como argumento el NID de la sección.
Editar traza	**URL**	**pantalasa-cms/edit-trace/%/%/%**
	Formulario:	SI
	Implementación:	pantalasa_cms_create_trace_form
	Acceso:	pantalasa_cms_edit_traces_ac
	Descripción:	Abre un formulario de edición de trazas. Recibe como argumentos el NID del requisito origen de la traza, el NID del proyecto y el NID de la traza.
Editar usuario	**URL**	**pantalasa-cms/edit-user/%**
	Formulario:	SI
	Implementación:	pantalasa_cms_create_user_form
	Acceso:	pantalasa_cms_create_edit_user_ac
	Descripción:	Abre un formulario para la edición de usuarios. Como argumento recibe el UID del usuario a modificar.
Eliminar catálogo	**URL**	**pantalasa-cms/delete-catalogo/%**
	Formulario:	NO
	Implementación:	pantalasa_cms_delete_catalogo
	Acceso:	pantalasa_cms_delete_catalog_ac
	Descripción:	Elimina un catálogo de requisitos. Recibe como argumento el NID del catálogo a eliminar.
Eliminar documento	**URL**	**pantalasa-cms/delete-documento/%**
	Formulario:	NO
	Implementación:	pantalasa_cms_delete_documento
	Acceso:	pantalasa_cms_delete_document_ac
	Descripción:	Elimina un documento de requisitos. Recibe como argumento el NID del documento a eliminar.
Eliminar hilo de discusión	**URL**	**pantalasa-cms/delete-thread/%/%/%**
	Formulario:	NO
	Implementación:	pantalasa_cms_delete_thread
	Acceso:	pantalasa_cms_delete_thread_ac
	Descripción:	Elimina un hilo de discusión. Recibe como

Operación		
		argumentos el NID del requisito, el NID del proyecto al que pertenece el requisito y el NID del propio hilo.
Eliminar miembros	**URL**	**pantalasa-cms/delete-member/%**
	Formulario:	SI
	Implementación:	pantalasa_cms_delete_member_from
	Acceso:	pantalasa_cms_delete_member_ac
	Descripción:	Abre un formulario para eliminar miembros de un grupo. Recibe como argumento el NID del grupo.
Eliminar proyecto	**URL**	**pantalasa-cms/delete-proyecto/%**
	Formulario:	NO
	Implementación:	pantalasa_cms_delete_proyecto
	Acceso:	pantalasa_cms_delete_proyecto_ac
	Descripción:	Elimina un proyecto. Recibe como argumento el NID del proyecto a eliminar.
Eliminar requisito	**URL**	**pantalasa-cms/delete-requirement/%**
	Formulario:	NO
	Implementación:	pantalasa_cms_delete_requirement
	Acceso:	pantalasa_cms_check_permissions_requirement
	Descripción:	Elimina un requisito. Recibe como argumento el NID del requisito a eliminar.
Eliminar sección	**URL**	**pantalasa-cms/delete-seccion/%**
	Formulario:	NO
	Implementación:	pantalasa_cms_delete_seccion
	Acceso:	pantalasa_cms_delete_section_ac
	Descripción:	Elimina una sección de requisitos. Recibe como argumento el NID de la sección a eliminar.
Establecer coordinador	**URL**	**pantalasa-cms/set-coord/%**
	Formulario:	SI
	Implementación:	pantalasa_cms_set_proy_coord_form
	Acceso:	pantalasa_cms_set_proy_coord_ac
	Descripción:	Abre un formulario para seleccionar el coordinador del proyecto. Como argumento recibe el NID del proyecto.
Gestionar parámetros	**URL**	**pantalasa-cms/manage-params/%/%**
	Formulario:	SI
	Implementación:	pantalasa_cms_manage_params_form
	Acceso:	pantalasa_cms_manage_param_ac
	Descripción:	Abre un formulario para gestionar los parámetros de un requisito. Receibe como argumentos el NID del requisito y el NID del catálogo o proyecto donde está contenido.
Mover requisito a documento	**URL**	**pantalasa-cms/move-req-to-doc/%/%/%**
	Formulario:	NO
	Implementación:	pantalasa_cms_move_req_to_doc
	Acceso:	pantalasa_cms_move_req_to_doc_ac

Operación		
	Descripción:	Mueve un requisito que está en un catálogo o proyecto sin clasificar en documento a un documento concreto del catálogo o proyecto. Recibe como argumentos el NID del requisito a mover, el NID del documento destino y el NID del catálogo o proyecto donde está el requisito.
Mover requisito a sección	**URL**	**pantalasa-cms/move-req-to-sec/%/%/%**
	Formulario:	NO
	Implementación:	pantalasa_cms_move_req_to_sec
	Acceso:	pantalasa_cms_move_req_to_sec_ac
	Descripción:	Mueve un requisito que está en un documento sin clasificar en sección a una sección concreta del documento. Recibe como argumentos el NID del requisito a mover, el NID de la sección destino y el NID del documento donde está el requisito.
Reusar catálogo	**URL**	**pantalasa-cms/reuse-catalog/%/%**
	Formulario:	NO
	Implementación:	pantalasa_cms_reuse_catalog
	Acceso:	pantalasa_cms_reuse_catalog_ac
	Descripción:	Copia la estructura de un catálogo y añade su contenido en un proyecto. Recibe como argumentos el NID del catálogo y el NID del proyecto.

Tabla 29 - Opciones de menú y navegación

5.4.4 Roles

Drupal proporciona flexibilidad a la hora de gestionar usuarios y roles de usuario. Permite definir un conjunto de roles donde cada uno de ellos tiene ciertos permisos que el administrador le ha concedido.

Habrá roles que podrán crear contenido, otros que sólo podrán consultar contenido, otros que puedan gestionarlo todo, etc. Todo depende de las necesidades de nuestro sitio web.

Por otro lado tenemos a los usuarios que son instancias concretas de un rol, es decir, un usuario pertenece a un rol y un puede haber varios usuarios con un mismo rol.

Esto permite otorgar a un conjunto de usuarios permisos comunes bajo un rol de forma sencilla ya que solamente hemos de indicar a qué rol pertenece un usuario concreto y estos usuarios heredarán todos sus permisos.

Drupal cuenta con tres roles por defecto: el rol administrador, usuario autenticado y usuario anónimo.

El rol administrador puede hacerlo todo en el sistema. Puede crear contenido, administrar la estructura, los tipos de contenido, vistas, etc. Es un rol con permisos para gestionar todo el sistema.

El usuario autenticado es un rol que generalmente tiene unos limitados a la consulta de contenido y opciones limitadas de gestión como crear contenido y administrar el contenido propio. Todo esto depende del sitio web.

Por último, Drupal permite definir los permisos para usuarios anónimos, es decir, visitantes de nuestro sitio web que no se autentican en el sistema. Principalmente realizarán operaciones de consulta limitadas.

En nuestra herramienta contaremos con cinco roles más: responsable del repositorio, coordinador, moderador, representante de equipo y analista.

5.4.4.1 Responsable del repositorio

Debemos dejar claro que el responsable del repositorio no tiene nada que ver con el administrador de Drupal.

El responsable del repositorio tiene tareas generales de administración y permisos especiales para gestionar cierto tipo de contenido que no competen a otros roles.

Sus tareas son las siguientes:

- Gestionar dominios
- Gestionar catálogos
- Gestionar tipos de datos
- Gestionar coordinadores
- Crear proyectos
- Asignar coordinador a proyecto

Tiene permisos para crear catálogos de requisitos, crear documentos, secciones y requisitos, permisos para poder modificar cualquier catálogo y también para eliminarlo.

Puede añadir y eliminar dominios y perfiles así como añadir y eliminar los tipos de datos para los parámetros de los requisitos parametrizables.

Sus permisos se limitan a la gestión de catálogo propiamente dicho. No tiene competencia sobre proyectos salvo para crearlos y asignarle un coordinador, por tanto también se encarga de gestionar a los usuarios coordinadores del repositorio.

Los permisos se van a gestionar mediante funciones propias pero se da un caso en el que podemos aprovechar la gestión que hace Drupal sobre el contenido. Es el caso de los tipos de parámetros. Será sólo el responsable del repositorio quien se encargue de gestionar estos contenidos y dado que sólo habrá un responsable del repositorio podemos asignar los permisos que establece Drupal para creare, editar y eliminar estos contenidos junto con los formularios estándar que proporciona.

5.4.4.2 Coordinador

El rol coordinador tiene designadas las funciones de poder editar un proyecto, aunque no puede crearlo. La edición se basa en el nombre y descripción del proyecto así como crear usuarios y grupos.

Se encarga de crear a todo tipo de usuarios involucrados en el proyecto y formar grupos de trabajo.

También puede consultar los documentos y requisitos de un proyecto pero no puede crearlos. Tan solo puede crear una versión base de un documento. Estas versiones base no son modificables.

En cuanto a permisos hay que decir que serán gestionados mediante funciones propias salvo por la gestión de grupos. Sólo el coordinador puede crear grupos, editarlos y eliminarlos pero no todos los grupos sino sólo aquellos que haya creado. Drupal permite asignar un permiso a este rol, el de eliminar contenido propio del tipo grupo. Para la creación y edición de grupos usaremos un formulario propio.

5.4.4.3 Moderador

El usuario moderador puede gestionar los proyectos creando documentos, secciones y requisito así como reusar catálogos de requisitos. Esta funcionalidad también la tienen el representante de equipo y el analista de forma que hay muchas similitudes en cuanto a funcionalidad con estos tres roles.

La principal diferencia entre ellos es que el moderador puede crear hilos de discusión y los analistas y representantes sólo puede añadir comentarios a esos hilos.

5.4.4.4 Representante de Equipo / Analista

A nivel funciona de esta herramienta ambos roles no presentan diferencias pero conviene implementarlos por separado para facilitar la ampliación futura de la misma.

La única distinción que se hace entre ambos es que el tipo de contenido grupo tiene una referencia a un usuario de tipo representante de equipo.

5.4.4.5 Usuario / Usuario Clave

Estos tipos de usuario no tiene funcionalidad alguna en la herramienta salvo para consultar información de los proyectos a los que pertenecen.

Su funcionalidad en esta herramienta es la misma pero ambos roles se implementarán por separado para facilitar las futuras ampliaciones de la herramienta. En la Tabla 30 se muestra un resumen de los roles y permisos que se usan de los que disponibles en Drupal:

5.4.4.6 Resumen de roles y permisos

Permisos/Rol	Roles	Permiso Drupal
Crear tipo de dato	• Responsable del repositorio	create parametro_type content
Modificar tipo de dato (propio)	• Responsable del repositorio	edit own parametro_type content
Modificar tipo de dato	• Responsable del	edit any parámetro_type

(cualquiera)	repositorio	content
Eliminar tipo de dato (propio)	• Responsable del repositorio	delete own parámetro_type content
Eliminar tipo de dato (cualquiera)	• Responsable del repositorio	delete any parámetro_type content

Tabla 30 - Resumen de roles y permisos de Drupal usado

5.4.5 Menús

Con el módulo menú vamos a definir cuatro menús para cuatro grupos de usuarios.

Por un lado tenemos al responsable de repositorio que de acuerdo a sus tareas se le proporcionarán enlaces de administración y gestión de catálogos, tipos de parámetros y usuarios coordinadores.

Seguimos con el rol coordinador el cual tiene competencias de acceso a proyectos, a la creación de usuarios sobre sus proyectos y a la creación de grupos con los usuarios de sus proyectos.

Luego tenemos al analista, moderador y representante los cuales tienen ciertas similitudes en cuanto a funcionalidad. Los tres roles pueden gestionar sus proyectos, añadir valores a un tipo de parámetro y consultar los catálogos.

En una cuarta clasificación tenemos a usuarios y usuarios clave que sólo podrán consultar los catálogos y sus proyectos.

La creación de menús en Drupal se hace con la función menu_save() que recibe como parámetro un array con la información del menú. La información que nosotros usaremos será el nombre máquina del menú para referenciarlo internamente, un título y una descripción.

La creación se realiza de la siguiente forma:

```
$menu = array(
      'menu_name' => 'nombre_maquina_del_menu',
      'title' => 'título del menú',
      'description' => 'descripción textual del menú',
);
menu_save($menu);
```

Ahora se crean los enlaces donde cada enlace es un array y se usa la función menu_link_save() que recibe como parámetro el array que representa el enlace. La información básica de un enlace para su creación es la URL a la que lleva, el texto del enlace, el menú al que pertenece y su posición en el listado de enlaces del menú.

Nota: antes de crear los enlaces del menú conviene limpiar la caché de Drupal para evitar inconsistencias.

Un enlace de menú se crea de la siguiente forma:

```
$link = array(
      'link_path' => 'URL del enlace',
      'link_title' => 'texto del enlace',
      'menu_name' => 'nombre máquina del menú'
      'weight' => 'peso. A mayor peso más abajo aparece',
      'expanded' => '0 / 1 indica si un enlace es desplegable',
);
```

```
menu_link_save($link);
```

Con ese código se crean los menús y los enlaces pero no se ha dicho nada de permisos y roles. Esto es debido a que no se pueden configurar los permisos deseados sobre un menú pero si sobre el bloque que lo contendrá. Aquí aparece la función nombrada anteriormente 'pantalasa_cms_actualizar_menus'.

Esta función activa el bloque del menú y le asigna los permisos al bloque. Para activar un bloque hemos programado una función genérica llamada "pantalasa_cms_activate_block".

Esta función está definida en el fichero /functions/commons.inc y recibe como parámetros el nombre del módulo, en nuestro caso "menu", el nombre máquina del bloque, que en nuestro caso es el mismo que el del menú, el nombre de la región donde mostrar el bloque, el theme para el que estará disponible, las páginas donde se mostrará el bloque y si está o no visible.

Realiza una actualización de la tabla "block" de la base de datos de Drupal con los parámetros pasados.

Una vez se ha activado el bloque le asignamos permisos a nivel de rol. Estos permisos son guardados en la tabla 'block_role' de la base de datos de Drupal por lo que debemos hacer un insert de los campos necesarios. Esos campos son el nombre del módulo, en nuestro caso "menu", el nombre del bloque, en nuestro caso el nombre del menú y el role id del role que tiene acceso al menú. Para ello necesitamos recuperar el objeto role del correspondiente rol.

El código sería el siguiente:

```
$role = user_role_load_by_name('nombre_del_role');
db_insert('block_role')
->fields(array(
      'module' => 'nombre del módulo',
      'delta' => 'nombre del bloque',
      'rid' => $role->rid,
))
->execute();
```

Con esta API que proporciona Drupal se inserta un registro fácilmente en la tabla. La función user_role_load_by_name() pertenece al módulo "user" y recibe como parámetro el nombre máquina del rol.

Llevamos a cabo esta acción para los cuatros menús. Para el menú analista al cual tendrán acceso tres roles haremos tres inserts.

A continuación se describen los cuatro menús:

5.4.5.1 Menú Responsable del repositorio

Los enlaces de navegación que tendrá este menú son los siguientes:

- Catalogs
- Projects
- Users

- Param types

El enlace “Catalogs” lleva a una vista bajo la URL catalogos-repositorio-gestion. Esta vista proporciona la información y enlaces a tareas propias de administración de catálogos.

El enlace “Projects” nos lleva a la vista con URL proyectos-repositorio-admon que nos muestra un listado de proyectos con opciones permitidas al responsable del repositorio sobre un catálogo.

“Users” nos lleva a la vista que tiene la URL gest-users-res que nos permite gestionar a los usuarios coordinadores y crear nuevos usuarios.

Finalmente, “Param types” nos lleva a la vista con URL tipos-parametros-gestion que nos muestra un listado de tipos de parámetros con opciones de gestión.

La implementación de la creación de este menú se encuentra en el fichero /menu/menu-roles/responsable.inc y con nombre máquina ‘menu-responsable.

5.4.5.2 Menú Coordinador

Los enlaces de navegación que tendrá este menú son los siguientes:

- Projects
- Users
- Groups

“Projects” nos lleva a la vista con la URL “listado-proyectos-edicion” que muestra un listado de proyectos sobre a los que el usuario coordinador pertenece con un conjunto de operaciones posibles sobre esos proyectos.

“Users” abre una vista con URL “usuarios” que permite crear nuevos usuarios y asignarlos a un proyecto.

“Groups” lleva a la vista bajo la URL “grupos-coord” que permite gestionar grupos de usuarios.

La implementación de la creación de este menú se encuentra en el fichero /menu/menu-roles/coordinador.inc y con nombre máquina ‘menu-coordinador’.

5.4.5.3 Menú Analista

Este menú sirve para usuarios analista, moderador y representantes de equipo pero por simplicidad le asignamos este nombre.

Los enlaces que proporciona son los siguientes:

- Catalogs
- Projects
- Param types

"Catalogs" nos lleva a "catalogos-repositorio-query". Una vista que sólo muestra información de los catálogos del repositirio.

"Projects" nos dirige a "proyectos-repositorio-gestion". Esta vista muestra opciones de gestión de un proyecto sobre los proyectos a los que el usuario pertenece.

"Param types" nos lleva la vista con la URL "parametros-repositorio" que muestra un listado de tipos de parámetros existentes y nos da opciones de añadir nuevos valores.

La implementación de la creación de este menú se encuentra en el fichero /menu/menu-roles/analista.inc y con nombre máquina 'menu-analista'.

5.4.5.4 Menú Usuario

Los enlaces que proporciona este menú son:

- Catalogs
- Projects

"Catalogs" nos lleva a la vista de consulta de los catálogos del repositorio, "catalogos-repositorio-query".

"Projects" nos dirige a una vista de consulta de proyectos sobre los que el usuario tiene acceso, "proyectos-repositorio-query".

La implementación de la creación de este menú se encuentra en el fichero /menu/menu-roles/usuario.inc y con nombre máquina 'menu-usuario'.

5.4.6 Vistas

Las vistas son consultas especializadas que recuperan los datos deseados y los formatean para que sean visibles al usuario de forma agradable, cómoda e intuitiva.

Contamos con el móduvo Views y Views UI que permite de forma gráfica crear vistas avanzadas y muy personalizables para mostrar el contenido del repositorio a los distintos usuarios.

Estas vistas se crean de forma gráfica de una manera muy sencilla pero necesitamos que esas vistas se generen automáticamente en el momento de la instalación del módulo. El módulo views permite exportar el código PHP de la vista y guardarlo en un fichero. De esta forma, en el momento de la instalación solamente necesitamos decirle a Drupal mediante hooks, dónde buscar las vistas por defecto del módulo.

Con el hook_views_api indicamos a Drupal la versión de Views que estamos usando, en nuestro caso la 3.0. Con el hook_views_default_views indicamos a Drupal cuáles son el código de las vistas por defecto. En nuestro caso tenemos el código de cada vista en un fichero. Lo que hacemos es incluir esos fichero dentro de la función y añadir la vista al array $views. Finalmente se retorna ese array.

Una vista puede tener varios "displays", que son variaciones entre una vista y otra. Por ejemplo, crearemos una vista para listar los catálogo y crearemos dos displays en la vista, un

display para el responsable del repositorio, con más opciones e información y otro displays para otros usuarios con opciones solamente de consulta.

Cada uno de los displays tiene una URL distinta y representarte de distinta forma ya que no siempre un display será un página, también puede ser un bloque que podemos embeber en cualquier página.

Podemos añadir o quitar campos de tipos de contenidos a las vistas e incluso hacer joins con tablas de varios tipos de contenido y mostrar información de todos esos tipos. También es posible crear campo personalizados como tipo texto o enlaces. Además tenemos campos ya creados como puede ser la fecha de creación, publicación, edición, usuario que lo creó, etc.

Podemos establecer criterios de ordenación con esos campos, añadir filtros a la consulta y establecer distintas formas de representación: representación por defecto, en forma de lista, en forma de tabla, etc.

Es posible añadir argumentos a la consulta de forma que podemos filtrar la consulta de forma dinámica a modo de argumento a través de la URL, argumento generado de forma aleatoria, etc.

Además, el módulo Views permite asignar comportamiento cuando se omite algún argumento o no hay información dando distintas opciones como mostrar una página de acceso denegado, proporcinar un resumen de los resultados, dar un valor por defecto, etc.

Para facilitar la navegación por los resultados, el módulo Views permite paginar los resultados en el número de resultados que deseemos. También debemos mencionar que permite restrigir el acceso por roles, por creación de contenido, etc.

Este módulo es muy potente y presenta gran cantidad y variedad de opciones de personalización que nos llevaría mucho tiempo mencionar sin profundizar demasiado. Aquí se ha hecho alusión a las que se han usado y más adelante, que se describa una vista en la que se haga uso de una cierta característica se hará un comentario más profundo de la misma.

A continuación vamos a describir los detalles más importantes de cada una de las vistas:

5.4.6.1 Listado de catálogos del repositorio (gestión)

Esta vista consta de dos displays. Un diplay será para mostrar los catálogos con opciones de creación y edición u otro display sólo tendrá opciones de consulta.

La vista tendrá la siguiente información:

Nombre: listado de catálogos del repositorio gestión	
Nombre máquina	vista_catalogos
Descripción	Listado de catálogos con opciones de gestión.

La información que vamos a mostrar es un listado en forma de tabla de todos los catálogos del repositorio de modo que en “format” estableremos el valor a “Table”.

5.4.6.1.1 Display general

Los campos que vamos a mostrar son el título del catálogo y una serie de operaciones. Queremos que el campo título nos lleve a otra vista que nos abra el catálogo. Esto hay que tenerlo en cuenta a la hora de añadir el campo ya que habrá que pasar el identificador del catálogo seleccionado como argumento de la vista que lo abre por lo que necesitamos recuperar su NID (node Id). Para ello añadiremos un campo ya creado en Drupal llamado "Content NID" que nos muestra el node Id del nodo. No nos interesa mostrarlo por lo que podemos ocultarlo o como lo llama el módulo Views, "excluirlo del display". Como va a ser usado por el campo título lo colocamos en primer lugar aunque no se muestre ya que si no lo colocamos antes de definir el campo que lo va a usar no será posible usarlo.

Ahora añadimos el campo título, también creado por Drupal como "Content title". Por defecto abre el nodo mostrado de la forma por defecto que tiene Drupal. Si queremos que sea un enlace a otra vista debemos de abrir el menú de "rewrite results" y seleccionar la opción de "Output this field as a link". En el campo de texto pondremos la ruta de la vista a la que se redirigirá, en este caso, "abrir-catalogo-gestion". Como necesitamos pasarle como parámetro el NID del catálogo seleccionado usaremos los "Replacement patterns" de la vista donde tenemos uno llamado [nid]. La url quedaría así "abrir-catalogo-gestion/[nid]". Drupal sustituirá [nid] por el NID del catálogo seleccionado.

El campo que muestra las operaciones será un campo creado por nosotros, llamado "Custom field", pero antes de crear ese custom field vamos a crear un "Custom field" por cada una de las operaciones que vamos a proporcionar.

Por cada operación creamos un "Custom field" que no mostraremos en la vista directamente de modo que tenemos que seleccionar la opción "exclude from display". Añadimos el texto que queramos mostrar en el campo "Text" y en "Rewrite results" seleccionamos la opción "Output this field as a link" y escribimos la ruta que deseemos. También tenemos disponibles los "Replacement patterns".

Hacemos lo mismo para las cuatro operaciones las cuales mostraremos en el último campo. Este último campo tendrá como valor del campo "Text" los valores disponibles en "Replacement patterns" de los campos creados, por defecto [nothing_X]. Este campo si lo mostraremos en el display bajo el título "Operaciones".

Queremos además proporcionar una opción de creación de catálogo antes de mostrar el listado. Para hello, en la sección "Header" de la vista vamos a añadir un área de texto global que gracias al módulo "PHP filter" podemos escribir código PHP que se ejecutará. El código es una comprobación de rol e imprimir un enlace HTML a la opción de menú que abre el formulario de creación:

```
<?php
global $user;
if(isRole($user,'responsable_repositorio')){
      $url = url('pantalasa-cms/create-catalog');
      echo '<a href="'.$url.'">Create catalog</a>';
}
```

?>

Nombre: Catálogos repositorio	
Nombre máquina	page_1
Descripción	Listado de catálogos con opciones de gestión.
Path	catalogos-repositorio-gestion
Parámetros	
Operaciones	• Add document (pantalasa-cms/create-document/[nid]) • Add requisite (pantalasa-cms/create-requirement/[nid]) • edit (pantalasa-cms/edit-catalog/[nid]) • delete (pantalasa-cms/delete-catalogo/[nid])
Roles	• Responsable del repositorio

5.4.6.1.2 Display query

Este display es similar al anterior. De hecho, es una copia del anterior al que quitamos los campos de operaciones y modificamos el enlace del título para que nos lleve a otra vista distinta donde poder abrir un catálogo en modo consulta, "documentos-catalogo-query/[nid]".

Tampoco se mostrará la cabecera de creación de nuevos catálogos en este display.

Nombre: Query	
Nombre máquina	vista_catalogos
Descripción	Listado de catálogos en modo consulta.
Path	catalogos-repositorio-query
Parámetros	
Operaciones	
Roles	• Roles PANTALASA_CMS

5.4.6.2 Listado de catálogos del repositorio (reúso)

Esta vista será visualizada al pulsar la opción de "reusar catálogo" sobre algún proyecto y mostrará un listado simple de los catálogos existentes en el repositorio con un enlace a una operación que implementa la reutilización de un catálogo concreto.

Será el campo "title" el que redefiniremos para que sea un enlace a una URL, en este caso a la URL "pantalasa-cms/reuse-catalog/[nid]/!1". Notar por la URL que hemos de añadir el campo "Content NID" antes del campo "title". El valor !1 es otro argumento que se recibe por la URL y que en esta vista no usaremos pero será neceario para la operación implementada en pantalasa-cms/reuse-catalog.

Añadiremos también un campo descripción a la vista para que muestre información del catálogo a reutilizar.

Como hemos dicho antes, esta vista recibirá un argumento necesario a través de la URL pero que no se utilizará en esta vista de modo que será un argumento de tipo "Null" que indica que no hará caso omiso del argumento aunque estará disponible en los "Replacement patterns".

Nombre: listado de catálogos del repositorio reúso	
Nombre máquina	Catalogos_reuso
Display	page

Descripción	Listado de catálogos para reusar.
Path	catalogos-reuso
Parámetros	NID proyecto
Operaciones	
Roles	• Moderador • Analista • Representante de equipo

5.4.6.3 Proyectos del repositorio (administración)

Esta vista muestra un listado de proyectos del repositorio con opciones de administración para el "Responsable del repositorio". Se añade en primer lugar una cabecera a la vista con un enlace a un formulario de creación de un nuevo proyecto escrito en PHP y usando el módulo PHP filter.

```
<?php
global $user;
if(isRole($user,'responsable_repositorio')){
      $url =url('pantalasa-cms/create-project');
      echo '<a href="'.$url.'">Create new project</a>';
}
?>
```

Añadimos el campo "Content NID" ya que vamos a necesitar pasar como argumento el NID del proyecto en la operación que se va a ofrecer. El campo "title" se deja tal cual lo crea el módulo "View" salvo porque se desactiva la opción de "Link this field to the original piece of content" para que no sea un enlace al nodo del proyecto. No será un enlace en general.

La operación que se va a ofrecer es la de establecer el usuario coordinador del proyecto de la misma forma que se han añadido las operaciones en vistas anteriores. Añadimos un "Custom field" que no se mostrará en el display con el texto "set coordinator" y que será un enlace a la URL "pantalasa-cms/set-coord/[nid]".

Crearemos otro "Custom field" con título "Operations" que haciendo uso de los "Replacement patterns" mostrará el campo anterior.

Nombre: proyectos del repositorio administración	
Nombre máquina	proyectos_del_repositorio_administraci_n_
Display	page
Descripción	Listado de proyectos con opciones de administración.
Path	proyectos-repositorio-admon
Parámetros	
Operaciones	• Set coordinator
Roles	• Responsable del repositorio

5.4.6.4 Proyectos del repositorio (edición)

Nombre: proyectos del repositorio edición	
Nombre máquina	listado_de_proyectos_edici_n_
Descripción	Listado de proyectos con opciones de edición.
Path	listado-proyectos-edicion
Parámetros	
Operaciones	• Edit • Delete
Roles	• Coordinador

5.4.6.5 Proyectos del repositorio (gestión)

Esta vista tendrá dos displays. Un display con opciones de gestión de proyectos para ciertos roles y otro display con opciones de consulta para todos los roles de PANTALASA_CMS.

Nombre: proyectos del repositorio gestión	
Nombre máquina	proyectos_repositorio_gestion
Descripción	Listado de proyectos con opciones de gestión.

5.4.6.5.1 Display general

Este display tendrá un campo "title" que será redefinido para que sea un enlace para visualizar el contenido del de proyecto, "documentos-proyecto-gestion/[nid]". Necesitamos el campo "Content NID" para su uso.

Se añaden tres campos más de tipo "Custom field" para las operaciones que se van a ofrecer siguiendo la forma que ya se ha usado anteriormente para este tipo de campos. Serán representados en un cuarto "Custom field" llamado "Operations".

Pero no podemos mostrar todos los proyectos del repositorio. Sólo debemos mostrar aquellos proyectos a los que el usuario autenticado pertenezca. Añadiremos por tanto una "Relationship" del tipo "Content: Proyectos (proyecto_usuario_field) – reverse" que relaciona el campo proyecto del tipo usuario con los proyectos. Ahora necesitamos filtrar al usuario que en nuestro caso será el usuario autenticado en ese momento. Para ello añadimos un "Contextual filter" de tipo usuario al que asociaremos la relación creada anteriormente. Por último ese filtro tendrá un valor por defecto que será el UID del usuario que acceda. Esto se hace mediante PHP con el siguiente código:

```
global $user;
return $user->uid;
```

De esta forma nos aseguramos que se muestren sólo los proyectos a los que pertenece el usuario.

Nombre: Proyectos repositorio	
Nombre máquina	page
Descripción	Listado de proyectos con opciones de gestión.
Path	proyectos-repositorio-gestion
Parámetros	
Operaciones	• Add document (pantalasa-cms/create-document/[nid])

	• Create requirement (pantalasa-cms/create-requirement/[nid]) • Reuse catalog (catalogos-reuso/[nid])
Roles	• Moderador • Analista • Representante de equipo

5.4.6.5.2 Display query

Este display es similar al anterior con la única salvedad de que no se mostrará ninguna operación y la URL del campo "title" será "documentos-proyecto-query/[nid]".

Nombre: Proyectos repositorio	
Nombre máquina	page_1
Descripción	Listado de proyectos con opciones de gestión.
Path	proyectos-repositorio-query
Parámetros	
Operaciones	
Roles	• Roles PANTALASA_CMS

5.4.6.6 Documentos de un catálogo (gestión)

Esta vista muestra los documentos creados en un catálogo con algunas operaciones de gestión disponibles para el "Responsable del repositorio" y otras opciones simples de consulta.

Nombre: documentos de un catálogo gestión	
Nombre máquina	documentos_de_un_catalogo_gesti_n_
Descripción	Listado de documentos con opciones de gestión.

5.4.6.6.1 Display general

Añadiremos en primer lugar el campo "Content NID" de la misma forma que en los casos anteriores y redefiniremos el campo "title" para que sea un enlace al contenido del documento: "secciones-de-un-documento/[nid]/!1".

El segundo parámetro que se pasa en el enlace anterior es el NID del catálogo que contiene el documento y que recibe esta vista.

Mostraremos también el tipo de documento haciendo uso del campo que representa esta propiedad y formateado como texto plano.

Crearemos además el campo "Operations" de la misma forma que en los casos anteriores.

Como pie de está vista mostraremos un campo que embebe una vista que muestra los requisitos sin clasificar en documentos de un catálogo.

Nombre: documentos de un catálogo gestión	
Nombre máquina	page_2
Descripción	Documentos de un catálogo con operaciones de gestión.
Path	abrir-catalogo-gestion
Parámetros	NID catálogo

Operaciones	• Add section • Add requirement • Edit • Delete
Roles	• Responsable del repositorio

5.4.6.6.2 Display query

Esta vista es similar a la anterior. Se eliminan las operaciones disponibles y la vista embebida será otra distinta con opciones sólo de consulta en los requisitos.

Nombre: documentos de un catálogo gestión	
Nombre máquina	page_1
Descripción	Documentos de un catálogo consulta.
Path	documentos-catalogo-query
Parámetros	NID catálogo
Operaciones	
Roles	• Roles PANTALASA_CMS

5.4.6.7 Documentos de un proyecto (coordinación)

Esta vista muestra un listado de documento de un proyecto con operaciones para el coordinador.

La única operación disponible será la de crear una versión base del documento por lo que aprovecharemos el mismo campo "Operations" para mostrarlo. La URL del enlace es "pantalasa-cms/create-base-version/[nid]"

Mostraremos el tipo de documento y el título lo redefiniremos para que sea un enlace a "secciones-de-un-documento/[nid]/!1". El segundo argumento el cual recibe como primero argumento esta vista es el NID del proyecto.

Se añade un filtro para que sólo se muestren los documentos en estado "Modifiable".

Añadiremos en el pie de la vista dos vistas embebidas: los requisitos sin clasificar en secciones y un listado de documento base.

Nombre: documentos de un proyecto coordinación	
Nombre máquina	documentos_proyecto_coordinacion
Display	page
Descripción	Listado de documentos de un proyecto con opciones de coordinación.
Path	documentos-proyecto-coordinacion
Parámetros	NID proyecto
Operaciones	• Créate base versión (pantalasa-cms/create-base-version/[nid])
Roles	• Coordinador

5.4.6.8 Documentos de un proyecto (gestión)

Esta vista es similar a la anterior. Nos limitaremos a citar los cambios.

5.4.6.8.1 Display general

Aquí cambiamos los enlaces de las operaciones y establecemos los correspondientes según cada operación.

Nombre: documentos de un proyecto gestión	
Nombre máquina	page
Descripción	Listado de documentos de un proyecto con opciones de gestión.
Path	documentos-proyecto-gestion
Parámetros	NID proyecto
Operaciones	• Add section (pantalasa-cms/add-seccion-documento/[nid]) • Add requirement (pantalasa-cms/create-requirement/[nid]) • Edit (pantalasa-cms/edit-document/!1/[nid]) • Delete (pantalasa-cms/delete-documento/[nid]) • Export (pantalasa-cms/generate-xml/[nid])
Roles	• Moderador • Analista • Representante de equipo

5.4.6.8.2 Display query

Nombre: documentos de un proyecto consulta	
Nombre máquina	page_1
Descripción	Listado de documentos de un proyecto con opciones de consulta.
Path	documentos-proyecto-query
Parámetros	NID proyecto
Operaciones	
Roles	• Roles PANTALASA_CMS

5.4.6.9 Documento creados en un proyecto

El propósito de esta vista es mostrar un listado de los documentos creado en un proyecto para mover requisitos de un proyecto a un documento en concreto.

Redefinimos el campo “title” con la URL de una operación que mueve los requisitos a un documento, “pantalasa-cms/move-req-to-doc/!1/[nid]/!2”. También reescribimos el texto para que muestre el tipo de documento y el título del documento en este campo. Necesitamos, por tanto, incluir el campo oculto de tipo de documento.

Añadiremos a la vista dos argumentos: el primero, que será de tipo Null ya que sólo usaremos para reenviarlo a la operación de mover requisitos, será el NID del requisito a mover. El segundo es el NID del proyecto que contiene el documento y este si se usa para mostrar los documento del proyecto.

Nombre: documentos de un proyecto	
Nombre máquina	documentos_creados_proyecto
Display	page
Descripción	Listado de documentos creados en un proyecto para mover requisitos.
Path	documentos-creados-proyecto
Parámetros	NID proyecto

Nombre: documentos de un proyecto	
Operaciones	
Roles	• Moderador • Analista • Representante de equipo

5.4.6.10 Documentos base

Hasta ahora todas las vistas creadas son de tipo "page" pero en este caso vamos a usar el tipo "block". La principal diferencia es que este tipo de vistas podemos incluirlas en otra páginas y posiciones donde se pueda ubicar un bloque. La implementación es igual que en las páginas.

Filtraremos los documentos a aquellos de tipo "Base" y también filtraremos los documentos mediante un argumento de la vista, el NID del proyecto.

Mostraremos el título del documento y el tipo además de la fecha de creación. La fecha será un criterio de ordenación. Este tipo de documentos serán de tipo consulta así que la vista llevará a otra vista bajo la URL "secciones-documento-query/[nid]/!1".

Nombre: secciones de un documento	
Nombre máquina	base_documents
Display	block_1
Descripción	Muestra los documentos base de un proyecto.
Path	
Parámetros	NID catálogo/proyecto
Operaciones	
Roles	• Roles PANTALASA_CMS

5.4.6.11 Secciones de un documento

Con esta vista mostraremos las secciones existentes en un documento con dos displays diferentes.

Nombre: Secciones de un documento gestión	
Nombre máquina	secciones_de_un_documento
Descripción	Listado de secciones de un documento con opciones de gestión.

5.4.6.11.1 Display general

Los campos a mostrar son el número de sección, el campo título y las operaciones. El campo título será reescrito para que nos lleve a la URL "requisitos-de-seccion/[nid]/!2" donde !2 es un argumento de la vista que tiene el valor del NID del proyecto.

Incluimos una vista embebida en el pie con los requisitos del documento.

Nombre: secciones de un documento	
Nombre máquina	secciones_de_un_documento
Display	page
Descripción	Muestra la secciones de un documento.
Path	secciones-de-un-documento
Parámetros	NID documento

Nombre: secciones de un documento	
	NID catálogo/proyecto
Operaciones	• Add requirement (pantalasa-cms/create-requirement/[nid]) • Edit (pantalasa-cms/edit-section/[nid]) • Delete (pantalasa-cms/delete-seccion/[nid])
Roles	• Responsable del repositorio • Moderador • Analista • Representante de equipo

5.4.6.11.2 Display query

La diferencia con el display anterior es que no ofrece operaciones y el título nos lleva a "requisitos-seccion-query/[nid]/!2".

Nombre: secciones de un documento	
Nombre máquina	page_1
Descripción	Muestra la secciones de un documento (consulta).
Path	secciones-documento-query
Parámetros	NID documento NID catálogo/proyecto
Operaciones	
Roles	• Roles PANTALASA_CMS

5.4.6.12 Subsecciones

Esta vista es muy similar a la de secciones de un documento salvo porque recibe como argumento de vista el NID de la sección padre y como URL del título nos lleva a la misma vista "rquisitos-de-subseccion/[nid]/!2".

Nombre: subsecciones	
Nombre máquina	subsections
Descripción	Listado de subsecciones de una sección.
Display general	block
Display consulta	block_1
Path	
Parámetros	NID documento
Operaciones	
Roles	• Responsable del repositorio • Moderador • Analista • Representante de equipo

5.4.6.13 Secciones creadas de un documento (mover requisitos)

Nombre: secciones creadas de un documento mover requisitos	
Nombre máquina	secciones_creadas_documento
Descripción	Listado de secciones creadas en un documento para mover requisitos.
Path	secciones-creadas-documento
Parámetros	NID documento

Nombre: secciones creadas de un documento mover requisitos	
Operaciones	
Roles	• Responsable del repositorio • Moderador • Analista • Representante de equipo

5.4.6.14 Requisitos

Las siguientes cuatro vistas tiene un formato muy similar. Son bloques que muestran los requisitos del nodo contenedor que se les pase como argumento. Sólo las describiremos brevemente ya que no presentan novedad respecto a vistas anteriores.

Añadir que cada una tiene un display general y un display de consulta (query).

5.4.6.14.1 Requisitos de un catálogo

Nombre: requisitos de un catálogo	
Nombre máquina	requisitos_de_un_catalogo
Display general	block
Display consulta	block_1
Descripción	Muestra un listado de los requisitos de un catálogo con opciones de gestión.
Path	
Parámetros	NID catálogo
Operaciones	• Param management (pantalasa-cms/manage-params/[nid]/!1) • Edit (documentos-creados-proyecto/[nid]/!1) • Move to document (documentos-creados-proyecto/[nid]/!1)
Roles	• Responsable del repositorio

5.4.6.14.2 Requisitos de un proyecto

Nombre: requisitos de un proyecto	
Nombre máquina	requisitos_de_un_proyecto
Display general	block
Display consulta	block_1
Descripción	Muestra un listado de los requisitos de un proyecto con opciones de gestión.
Path	
Parámetros	NID documento
Operaciones	• Param management (pantalasa-cms/manage-params/[nid]/!1) • Edit (pantalasa-cms/edit-requirement/!1/[nid]) • Move to document (documentos-creados-proyecto/[nid]/!1)
Roles	• Moderador • Analista • Representante de equipo

5.4.6.14.3 Requisitos de un documento

Nombre: requisitos de un documento	
Nombre máquina	requisitos_de_un_documento
Display general	block
Display consulta	block_1
Descripción	Muestra un listado de los requisitos de un documento con opciones de gestión.
Path	
Parámetros	NID documento
Operaciones	• Param management (pantalasa-cms/manage-params/[nid]/!1) • Edit (pantalasa-cms/edit-requirement/!1/[nid]) • Move to section (secciones-creadas-documento/[nid]/!1)
Roles	• Responsable del repositorio • Moderador • Analista • Representante de equipo

5.4.6.14.4 Requisitos de sección

Nombre: requisitos de sección	
Nombre máquina	requisito_de_secci_n
Display general	block
Display consulta	block_1
Descripción	Muestra un listado de requisitos de una sección con opciones de gestión.
Path	
Parámetros	NID sección NID catálogo/proyecto
Operaciones	• Param management (pantalasa-cms/manage-params/[nid]/!1) • Edit (pantalasa-cms/edit-requirement/!1/[nid]) • Delete (pantalasa-cms/delete-requirement/[nid])
Roles	• Responsable del repositorio • Moderador • Analista • Representante de equipo

5.4.6.15 Trazas de un requisito (gestión)

Esta vista consta de nuevo con dos displays: uno general y otro sólo para consultas.

Nombre: trazas de un requisito	
Nombre máquina	trazas_requisito_gestion
Descripción	Listado de trazas de un requisito con operaciones de gestión.

Estos displays son meramente informativos y muestran el nombre de la traza, el requisito destino y el tipo de traza. Añadir que aunque se pase como segundo argumento el NID del catálogo/proyecto no lo usaremos como filtro aunque será necesario para las operaciones.

5.4.6.15.1 Display general

Nombre: trazas de un requisito gestión	
Nombre máquina	page
Descripción	Listado de trazas de un requisito con operaciones de gestión.
Path	trazas-requisito-gestion
Parámetros	NID requisito NID catálogo/proyecto
Operaciones	• Edit (pantalasa-cms/edit-trace/!1/!2/[nid]) • Delete
Roles	• Moderador • Analista • Representante de equipo

5.4.6.15.2

5.4.6.15.3 Display query

Nombre: trazas de un requisito consulta	
Nombre máquina	pagea_1
Descripción	Listado de trazas de un requisito modo consulta.
Path	trazas-requisito-query
Parámetros	NID requisito NID catálogo/proyecto
Operaciones	
Roles	• Roles PANTALASA_CMS

5.4.6.16 Hilos de discusión (gestión)

Muestra un listado de hilos de discusión de un requisito. Las operaciones que ofrece son de edición y cerrado con información del título, descripción y estado.

Nombre: hilos de discusión gestión	
Nombre máquina	Hilos_discusion
Display	page
Descripción	Listado de tipos de hilos de discusión de un requisito con opciones de gestión.
Path	discussion-threads
Parámetros	NID requisito NID catálogo/proyecto
Operaciones	• Edit (pantalasa-cms/edit-discussion-thread/!1/[nid]) • Delete (pantalasa-cms/close-thread/!1/!2/[nid])
Roles	• Moderador • Analista • Representante de equipo

5.4.6.17 Tipos de parámetros (gestión)

Para esta vista y dado que sólo será usada por el responsable del repositorio los enlaces de edición y eliminación serán los propios que proporciona Drupal.

El único enlace personalizado que tendremos será el de creación que lo embeberemos dentro de un área de texto en la cabecera de la vista con el siguiente código PHP:

```
<?php
global $user;
if(isRole($user,'responsable_repositorio')){
      $url =url('node/add/parametro-type');
      echo '<a href="'.$url.'">Create new param type</a>';
}
?>
```

Nombre: tipos de parámetros gestión	
Nombre máquina	tipos_de_par_metros
Display	page
Descripción	Listado de tipos de parámetros con opciones de gestión.
Path	tipos-parametros-gestion
Parámetros	
Operaciones	• Edit • Delete
Roles	• Responsable del repositorio

5.4.6.18 Tipos de parámetos (uso)

Esta vista es un listado simple de los tipos de parámetros disponibles para mostrar los valores disopnibles. Para ello reescribiremos el campo título con un enlace, “valores-parametro/[nid]”.

Nombre: tipos de parámetros uso	
Nombre máquina	Tipos_de_parametro_uso
Display	page
Descripción	Listado de tipos de parámetros con opciones para usuarios.
Path	parametros-repositorio
Parámetros	
Operaciones	
Roles	• Moderador • Analista • Representante de equipo

5.4.6.19 Valores de parámetro

Esta vista muestra un listado de los valores disponibles para un tipo de parámetro cuyo NID se pasa como argumento de la vista. El contenido de esta vista es meramente informativo.

Nombre: valores de parámetro	
Nombre máquina	Valores_parametro
Display	page
Descripción	Listado de valores de un parámetro
Path	Valores-parametro
Parámetros	NID parámetro
Operaciones	
Roles	• Moderador

Nombre: valores de parámetro	
	• Analista • Representante de equipo

5.4.6.20 Usuarios

Para las siguientes dos vistas vamos a usar el mismo esquema. La única diferencia serán los roles que muestre. Añadimos el filtro de UID de usuario como argumento de la vista proporcionando un valor por defecto, el UID del usuario actual. Ese filtro tendrá como relación el campo "proyecto_usuario_field" del tipo Proyecto. También se añade la relación con el campo Proyectos del tipo Usuario.

De esta forma conseguimos que se muestren los usuarios de los proyectos del usuario actual.

Como cabecera añadimos un enlace de creación de usuarios que será un formulario personalizado.

5.4.6.20.1 Usuarios del responsable

Nombre: usuarios del responsable	
Nombre máquina	usuarios_responsable
Display	page
Descripción	Listado de usuarios gestionados por el responsable del repositorio.
Path	gest-user-res
Parámetros	• Usuario NID
Operaciones	• Edit
Roles	• Responsable del repositorio

5.4.6.20.2 Usuarios

Nombre: usuarios	
Nombre máquina	usuarios
Display	page
Descripción	Listado de usuarios creados en un proyecto con operaciones de gestión.
Path	usuarios
Parámetros	• Usuario NID
Operaciones	• Edit
Roles	• Coordinador

5.4.6.21 Grupos

Dado que los grupos serán solamente gestionados por el usuario coordinador crearemos una vista de un listado genérico de grupos con las operaciones que abajo se muestran.

Para que se muestren los grupos de un coordinador concreto añadiremos como filtro el UID del autor del contenido y dado que sólo el coordinador crea grupos ya tenemos el resultado deseado.

En enlace de "delete" será el propio de Drupal.

Nombre: grupos	
Nombre máquina	grupos

Nombre: grupos	
Display	page
Descripción	Listado de grupos creados en un proyecto con opciones de gestión.
Path	grupos-coord
Parámetros	• Author uid
Operaciones	• Add member (pantalasa-cms/add-member/[nid]) • Delete member (pantalasa-cms/delete-member/[nid]) • Edit (pantalasa-cms/edit-group/[nid]) • Delete
Roles	• Coordinador

5.4.7 Formularios

Para gestionar la creación y edición de contenido no podemos usar los formularios estándar que proporciona Drupal debido a que no es posible hacer una correcta gestión de los permisos de usuarios y roles con este sistema.

Drupal proporciona una API Form para crear formularios totalmente personalizados para limitar al usuario a la funcionalidad deseada y facilitarle la tarea.

Esas tareas son operaciones en el sistema que se invocan a través del menú del módulo. Para poder invocar a un formulario personalizado hemos de establecer en el "page callback" el valor de "drupal_get_form" que recibe como primer argumento el nombre de la función que implementa el formulario. Esa función retornará un array de nombre "form" que será una variable que representa el formulario. Cada posición del array es un campo del formulario.

Los formularios constan de una serie de campos de tipos variados, texto, lista, botones, etc., los cuales se representan como arrays en PHP, de forma que un formulario en Drupal es un array de arrays.

Para crear un campo lo que hacemos es instanciar una posición del array como otro array de la forma:

```
$form['field_name'] = array( … );
```

Las claves de ese array van precedidas del símbolo '#' para que sean identificadas como nombres en clave por Drupal. Vamos a mostrar un ejemplo general de creación de un campo del formulario:

```
$form['nombre'] = array(
      '#type' => 'textfield',
      '#title' => $t('Name'),
      '#weight' => 0,
      '#required' => TRUE,
);
```

Toda la información acerca de los tipos de campos y claves posibles la podemos encontrar en el sitio oficial [31].

Los formularios tendrán un botón de submit para enviar la información del formulario. Ese submit ha de implementarse con una función para manipular los datos del formulario. La referencia a esa función se establece al crear el botón de submit como clave #submit:

```
$form['submit']['#submit'] = array ('function_name');
```

De forma opcional se puede implementar una función para validar los datos del formulario antes de enviarlos al submit. La referencia a esta función se establece de igual modo en el submit como clave #validate:

```
$form['submit']['#validate'] = array('function_name');
```

A continuación se muestra un listado en la de las funciones que implementan formularios. Los ficheros de los formularios se encuentran el /menu:

Formulario		
Añadir miembro	**Principal**	**pantalasa_cms_add_member_form**
	Submit	pantalasa_cms_add_member_submit
	Validación	-
Crear catálogo	**Principal**	**pantalasa_cms_create_catalog_form**
	Submit	pantalasa_cms_create_catalog_submit
	Validación	-
Crear comentario hilo	**Principal**	**pantalasa_cms_add_comment_form**
	Submit	pantalasa_cms_add_comment_submit
	Validación	-
Crear documento	**Principal**	**pantalasa_cms_create_documento_form**
	Submit	pantalasa_cms_create_document_submit
	Validación	-
Crear grupo	**Principal**	**pantalasa_cms_gest_group_from**
	Submit	pantalasa_cms_gest_group_submit
	Validación	-
Crear hilo de discusión	**Principal**	**pantalasa_cms_create_discussion_thread_form**
	Submit	pantalasa_cms_create_discussion_thread_submit
	Validación	-
Crear proyecto	**Principal**	**pantalasa_cms_create_project_form**
	Submit	pantalasa_cms_create_project_submit
	Validación	-
Crear requisito	**Principal**	**pantalasa_cms_create_requisito_form**
	Submit	pantalasa_cms_create_req_submit
	Validación	-
Crear sección	**Principal**	**pantalasa_cms_create_seccion_form**
	Submit	pantalasa_cms_create_section_submit
	Validación	pantalasa_cms_create_edit_section_validate

Formulario		
Crear traza	**Principal**	**pantalasa_cms_create_trace_form**
	Submit	pantalasa_cms_create_trace_submit
	Validación	-
Crear usuario	**Principal**	**pantalasa_cms_create_user_form**
	Submit	pantalasa_cms_create_user_submit
	Validación	pantalasa_cms_create_user_validate
Crear valor de parámetro	**Principal**	**pantalasa_cms_create_param_value_form**
	Submit	pantalasa_cms_create_param_value_submit
	Validación	-
Editar catálogo	**Principal**	**pantalasa_cms_edit_catalog_form**
	Submit	pantalasa_cms_edit_catalog_submit
	Validación	-
Editar documento	**Principal**	**pantalasa_cms_create_documento_form**
	Submit	pantalasa_cms_create_document_submit
	Validación	-
Editar grupo	**Principal**	**pantalasa_cms_gest_group_form**
	Submit	pantalasa_cms_gest_group_submit
	Validación	-
Editar hilo de discusión	**Principal**	**pantalasa_cms_create_discussion_thread_form**
	Submit	pantalasa_cms_create_discussion_thread_submit
	Validación	-
Editar proyecto	**Principal**	**pantalasa_cms_create_project_form**
	Submit	pantalasa_cms_create_project_submit
	Validación	-
Editar requisito	**Principal**	**pantalasa_cms_create_requisito_form**
	Submit	pantalasa_cms_create_req_submit
	Validación	-
Editar sección	**Principal**	**pantalasa_cms_edit_section_form**
	Submit	pantalasa_cms_edit_section_submit
	Validación	pantalasa_cms_create_edit_section_validate
Editar traza	**Principal**	**pantalasa_cms_create_trace_form**
	Submit	pantalasa_cms_create_trace_submit
	Validación	-
Editar usuario	**Principal**	**pantalasa_cms_create_user_form**
	Submit	pantalasa_cms_edit_user_submit
	Validación	pantalasa_cms_edit_user_validate
Eliminar miembros	**Principal**	**pantalasa_cms_delete_member_form**
	Submit	pantalasa_cms_delete_member_submit
	Validación	-
Establecer coordinador	**Principal**	**pantalasa_cms_set_proy_coord_form**
	Submit	pantalasa_cms_set_proy_coord_submit
	Validación	-
Gestionar parámetros	**Principal**	**pantalasa_cms_manage_params_form**
	Submit	pantalasa_cms_manage_params_submit
	Validación	-

Formulario		
Gestionar patrones (subir)	**Principal**	**pantalasa_cms_manage_patterns_form**
	Submit	pantalasa_cms_upload_pattern_submit
	Validación	pantalasa_cms_upload_pattern_validate
Gestioanr patrones (eliminar)	**Principal**	**pantalasa_cms_manage_patterns_form**
	Submit	pantalasa_cms_delete_pattern_submit
	Validación	-
Búsqueda y reuso de requisitos	**Principal**	**pantalasa_cms_search_requirement_form**
	Submit	pantalasa_cms_search_requirement_submit
	Validación	-

Tabla 31 - Formularios y sus funciones

5.4.8 Bloques

Además de los bloques de contenido que se crean al crear vistas como bloques o al crear los menús de navegación vamos a crear explícitamente un bloque en nuestro módulo para que muestre en todo momento el rol del usuario logado de forma que si un usuario (persona física) tiene varios roles, sepa en todo momento con qué rol está trabajando en el sistema.

La forma de crear un bloque en Drupal es mediante el hook_block_info. Esta función define cada bloque como un array con una serie de valores. La forma general del crear un bloque es:

```
$blocks['block_name'] = array(
      'info' => t('Information'),
      'status' => block status,
      'region' => 'region_name',
);
return $blocks;
```

Esto crea el bloque pero no le asigna contenido. Para darle contenido al bloque hemos de implementar el hook_block_view($delta=''). En $delta viene el nombre del bloque que se quiere renderizar. Con un control de casos controlaremos qué bloque es y mediante:

```
$block['subject'] = '…';
$block['content'] = '…';
return $block;
```

Asignaremos el contenido del bloque que puede ser texto o código PHP.

Nuestro bloque tiene por nombre 'user_role' y está en la región 'header'. Como contenido del bloque mostramos un texto indicando el role al que pertenece.

Hemos añadido otro bloque para facilitar la navegación entre secciones de un documento. Este bloque sólo se muestra en las vistas de secciones y carga un formulario con un select con las secciones de ese documento. Al pulsar el submit nos redirige a la vista de la sección.

5.4.9 Otros

5.4.9.1 Funciones auxiliares

A lo largo de la implementación de la herramienta se ha ido refactorizando el código para mejorar la comprensión y aprovechar funcionalidad ya implementada. Se han detectado varias funciones interesantes que son usadas en varias partes del módulo y se han aglutinado en dos

ficheros. Uno de los ficheros es para funciones que tratan directamente con el contenido y otro fichero con otras funciones. Esos ficheros son /funtions/common.inc y /functions/content.inc. No vamos a detallar cada una de las funciones pero si a citar cómo se invocar y qué hacen:

5.4.9.1.1 Fichero db_functions.inc

En este fichero se proporcionan una serie de funciones para el acceso a base de datos para realizar consultas y eliminaciones de contenidos. Se muestran en la Tabla 32.

Función	Parámetros	Descripción
getInstances	• Id del requisito	Devuelve un array con las instancias de parámetros de un requisito.
getTraces	• Id del requisito	Devuelve un array con los NID de las trazas de un requisito
getThreads	• Id del requisito	Devuelve un array con los NID de los hilos de un requisito
getRequirementsSection	• Id de la sección	Devuelve un array con los NID de los requisitos de una sección.
getDocRequirementsNC	• Id del documento	Devuelve un array con los NID de los requisitos sin clasificar de un documento
getDocumentSections	• Id del documento	Devuelve un array con las secciones de un documento
getProRequirementsNC	• Id del proyecto	Devuelve un array con NID de los requisitos sin clasificar de un proyecto
getCatRequirementsNC	• Id del catálogo	Devuelve un array con los NID de los requisitos sin clasificar de un catálogo
getDocuments	• Id del catálogo o proyecto	Devuelve un array con los NID de los documentos de un proyecto o catálogo
getProjectGroups	• Id del proyecto	Devuelve un array con los NID de los grupos de un proyecto
deleteGroup	• Id del grupo	Elimina un grupo.
deleteParamInstance	• Id de la instancia	Elimina una instancia de un parámetro de un requisito
deleteTrace	• Id de la traza	Elimina una traza
deleteThread	• Id del hilo	Elimina un hilo
deleteRequirement	• Id del requisito	Elimina un requisto, sus trazas, sus instancias y sus hilos.
deleteSection	• Id de la sección	Elimina una sección.
deleteDocument	• Id del documento	Elimina un documento.
deleteProject	• Id del proyecto	Elimina un proyecto.

Función	Parámetros	Descripción
deleteCatalog	• Id del catálogo	Elimina un catálogo.

Tabla 32 - Funciones de db_functions.inc

5.4.9.1.2 Fichero common.inc

En la se muestra un listado de las funciones disponibles:

Función	Parámetros	Descripción
getRoleId	• Nombre del rol	Dado un nombre de rol devuelve el NID del rol.
isRole	• Objeto usuario • Nombre de rol	Devuelve TRUE si el usuario pasado pertenece a ese rol y FALSE en otro caso.
existsUser	• Nombre de usuario	Devuelve TRUE si existe un usuario con ese nombre y FALSE en caso contrario.
pantalasa_cms_activate_block	• Nombre del modulo • Nombre del bloque • Nombre de la región • Nombre del tema • Páginas • Visibilidad	Activa un bloque de un determinado módulo en una región concreta de un tema concreto para las páginas que se indican. También se indica si es visible o no.
getTerminoEnVocabulario	• Nombre del término • Nombre del vocabulario	Función que recupera el objeto término de un vocabulario dado el nombre del término y el del vocabulario.
getUsers	• NID proyecto	Devuelve todos los usuarios que pertenecen al proyecto en un array de objetos user.
buscarGrupos	• NID proyecto	Devuelve todos los grupos de un proyecto en un array de objetos grupo.
getProjects	• Objeto user	Devuelve todos los proyectos a los que pertenece un usuario en array de objetos proyecto.
isMember	• Objeto user • NID proyecto	Devuelve TRUE si el usuario pertenece al proyecto o FALSE en otro caso.
getIdTipoParametro	• Clave parámetro	Devuelve el NID el tipo de parámetro con dicha clave.
isPangeaRole	• Objeto usuario	Comprueba si el usuario es de un rol de pangea.

Tabla 33 - Funciones en common.inc

5.4.9.1.3 Fichero content.inc

En la Tabla 34 se muestra un listado de funciones:

Función	Parámetros	Descripción
copiarRequisito	• NID requisito	Hace una copia de un requisito y devuelve el nuevo requisito.
copiarSeccion	• NID sección	Hace una copia de la sección y devuelve la nueva.
generateId	• NID requisito • NID tipo contenedor	Devuelve el identificador que le corresponde a un requisito dado su NID y el NID del nodo que lo contiene.
getRequirements	• NID catálogo/proyecto	Devuelve un array con todos los requisitos de un proyecto cuyos índices son los NID y valor los identificadores.
Pertenece	• NID requisito • NID catálogo/proyecto	Devuelve TRUE si es requisito pertenece al catálogo/proyecto o FALSE en otro caso.
getContainer	• NID requisito	Devuelve el NID del contenido que contiene directamente al requisito.
generateTraceId	• NID requisito	Genera el identificador de traza de una trace. Recibe como parámetro el NID del requisito origen de la traza.
copiarTrazas	• NID requisito antiguo • NID requisito (copia) • Array de requisitos copiados	Copia las trazas de un requisito. Recibe como parámetros los NID del requisito copiado y de la copia y una tabla con todos los requisitos copiados asociando la clave, el NID del requisito copiado, y el valor, el NID de la copia del requisito.

Tabla 34 - Funciones de content.inc

5.4.9.2 Internacionalización

Parece lógico que la herramienta esté soportada en varios idiomas debido a que se trata de una herramienta para la gestión de requisitos en el desarrollo global de software.

Drupal facilita mucho el trabajo de crear interfaces en varios idiomas. Para traducir una interfaz basta con descargar el idioma deseado del sitio web de Drupal e instalarlo y la interfaz será traducida a dicho idioma. Esto se hace gracias a una función que proporciona Drupal, t.

Esta variable se recupera con la función t() y su uso consiste en introducir entre comillas simples el texto que quiere ser traducido.

En la herramienta se ha hecho uso de esta variable para mostrar todos los títulos, enlaces y nombres de forma que sea posible traducirlos.

A la hora de traducir la interfaz, Drupal buscará en el módulo las llamadas a esa función e intentará traducir el texto que encierra entre comillas simples.

5.4.9.3 Keywords

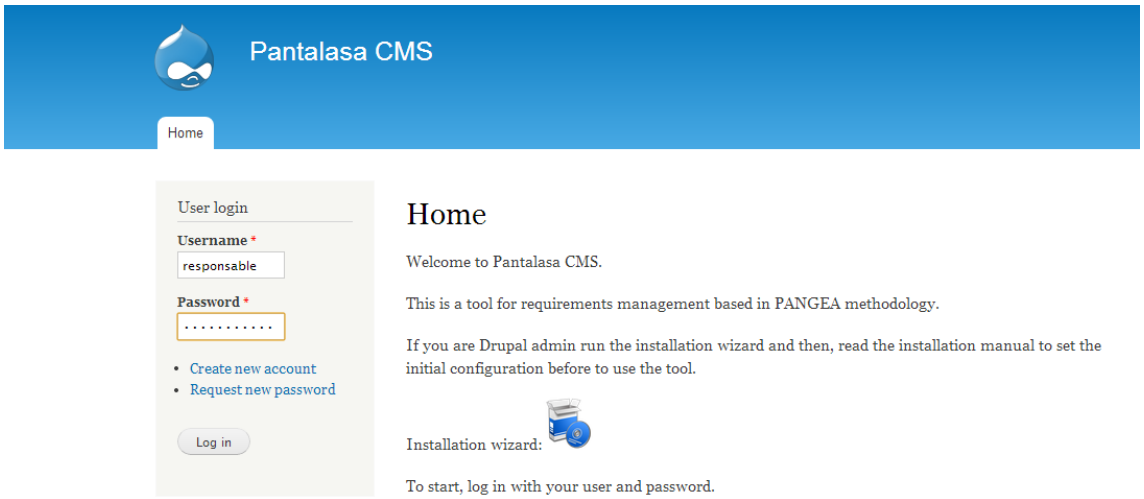
Para poder buscar de una forma sencilla los proyectos y requisitos hemos creado un campo que contiene palabras clave. Esas palabras clave se insertan en el momento de creación de un proyecto o un requisito y se almacenan en el repositorio para que estén disponibles para el resto de usuarios.

A la hora de mostrar los requisitos y proyectos en las vistas podemos aplicar un filtro que busque en el campo keywords el contenido deseado.

5.5 Validación

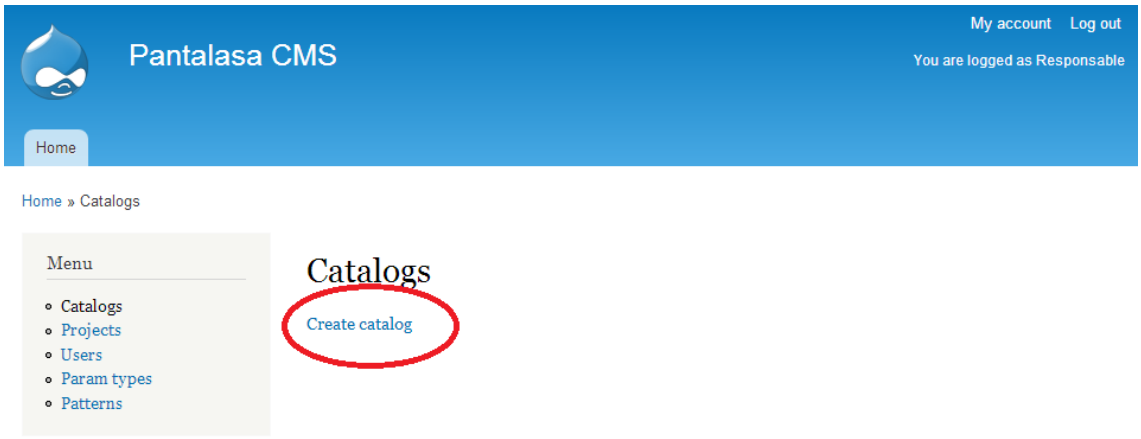
Para validar la herramienta vamos a crear un catálogo de internacionalización con un documento de requisitos.

Lo primero que hacemos es acceder a la herramienta. Para crear un catálogo debemos entrar con el usuario "responsable" que es quien tiene el rol de "Responsable del repositorio":

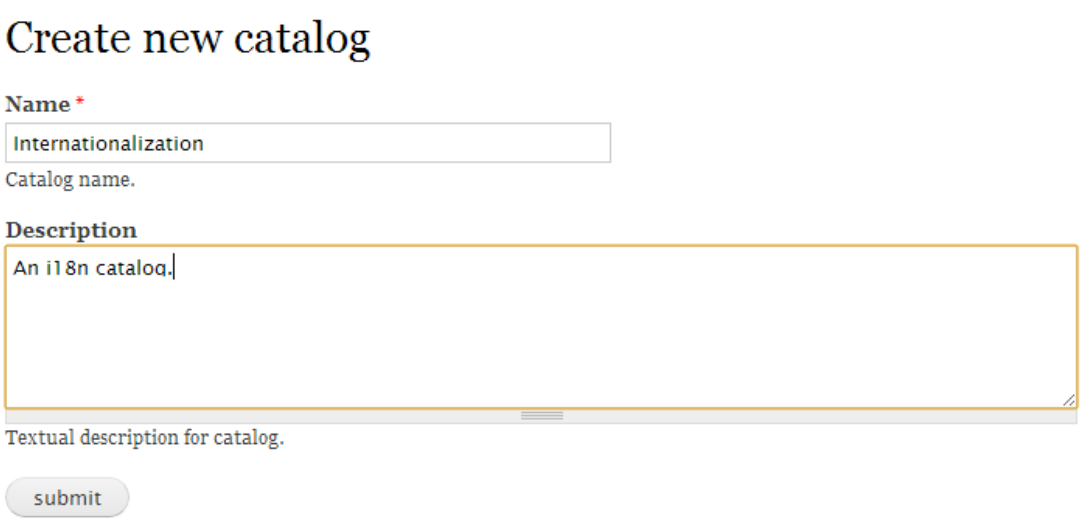


Al logarnos entramos a la página "Home" del responsable donde nos indica con qué rol nos hemos logado y nos muestra el menú de opciones para ese usuario:

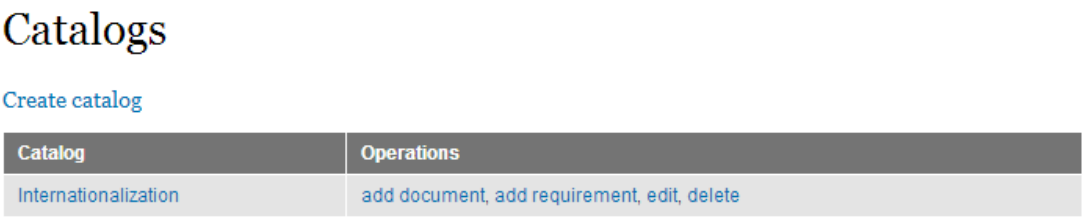
Vamos a crear un catálogo. Para ello hacemos click en "Catalogs" y se nos muestra la vista de catálogos y la opción de crear uno nuevo. Como no hay catálogos creados aun, aparece vacía:



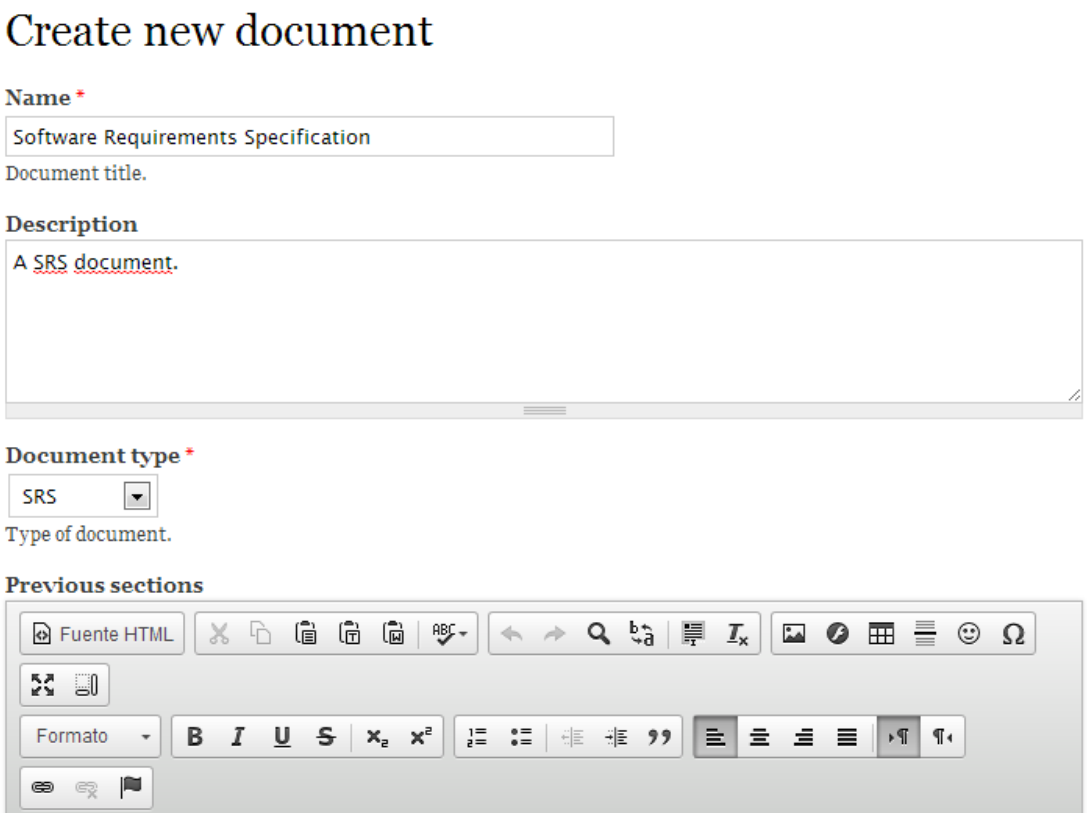
Hacemos click en “Create catalog” para abrir un formulario de creación de un catálogo:



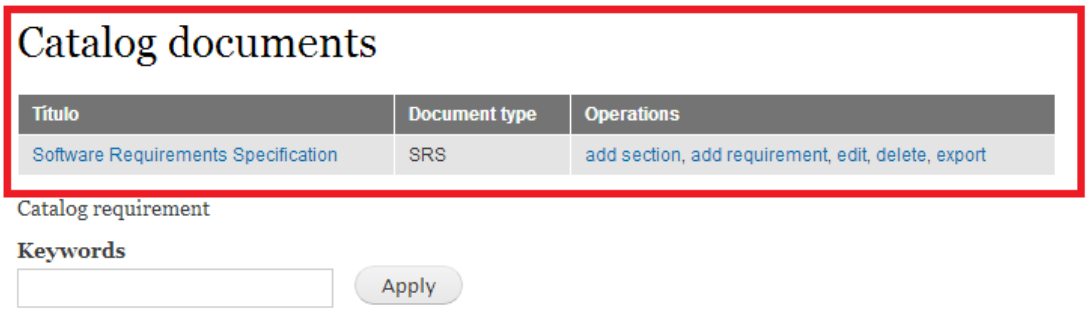
Pulsamos “submit” y se crea el catálogo. Si ahora vamos a la vista de catálogos nos aparecerá el catálogo creado:



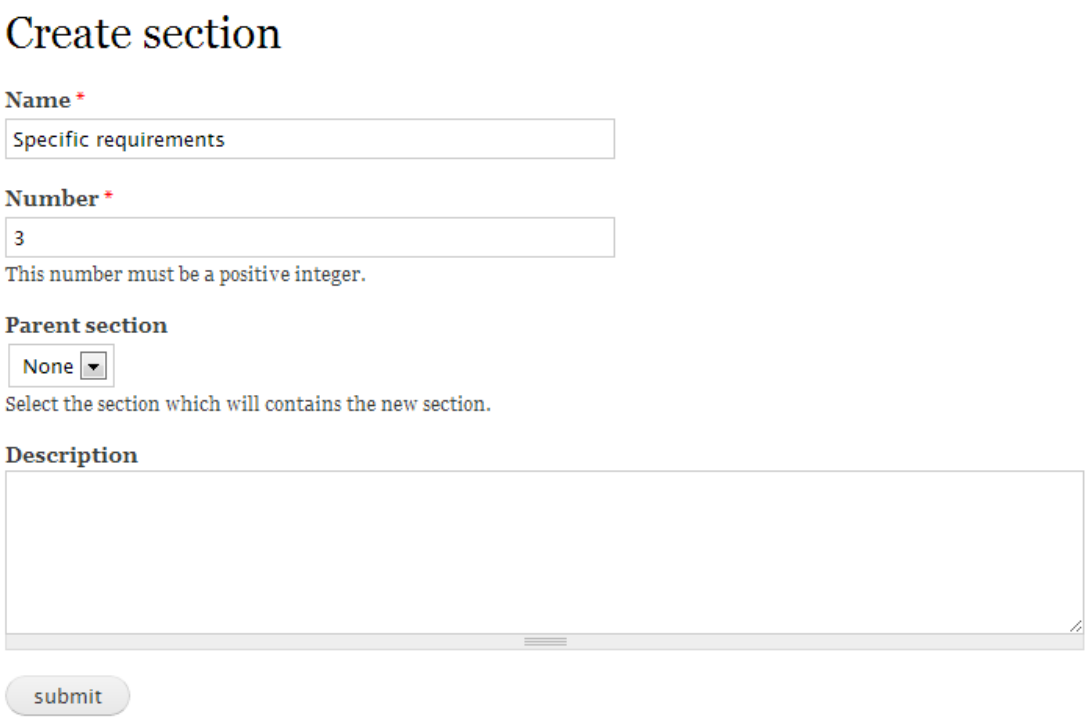
Ahora vamos a añadir un documento. Para ello hacemos click en “add document” y se nos abrirá un formulario de creación de documentos:



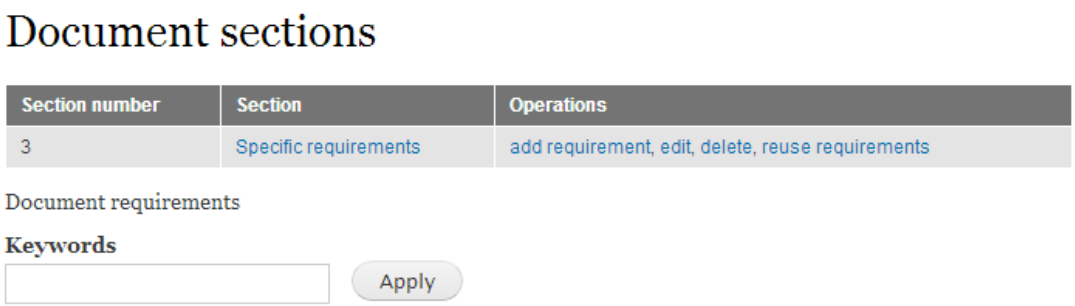
Hacemos click en "submit" en se creará el documento y automáticamente se añadirá al catálogo. Si nos vamos a la vista de catálogos y pulsamos sobre el nombre del catálogo se nos mostrará un listado con los documentos de ese catálogo:



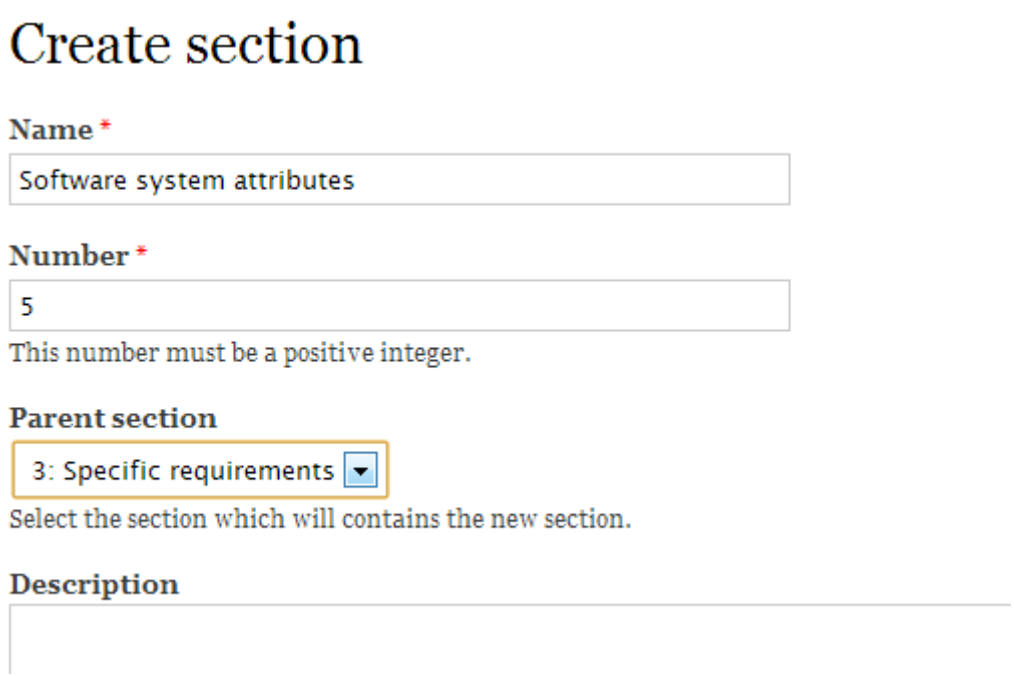
Además, en esta vista se muestra el listado de requisitos sin clasificar en documentos y un campo para filtrar requisitos por palabras clave. Ahora vamos a crear una sección del documento. La sección está numerada como 3 y tiene el nombre de "Specific Requirements". Añadimos la sección al documento haciendo click en "add section":



Como es una sección principal no establecemos ninguna sección padre. Hacemos click en "submit" y se creará la sección. Si nos vamos al listado de documentos del catálogo y hacemos click en el título del documento podemos ver el listado de secciones principales del documento:

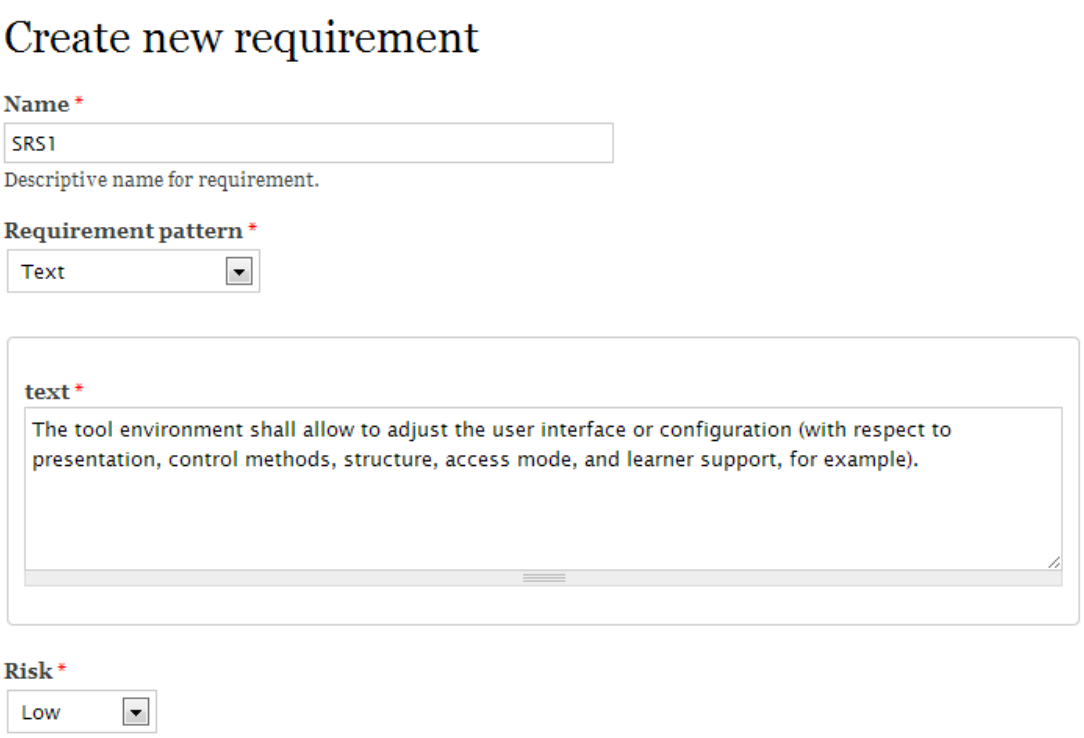


Ahora vamos a añadir una subsección. La forma de hacerlo es desde el listado de documentos, haciendo click en "add section" pero esta vez seleccionaremos como sección padre la sección 3:

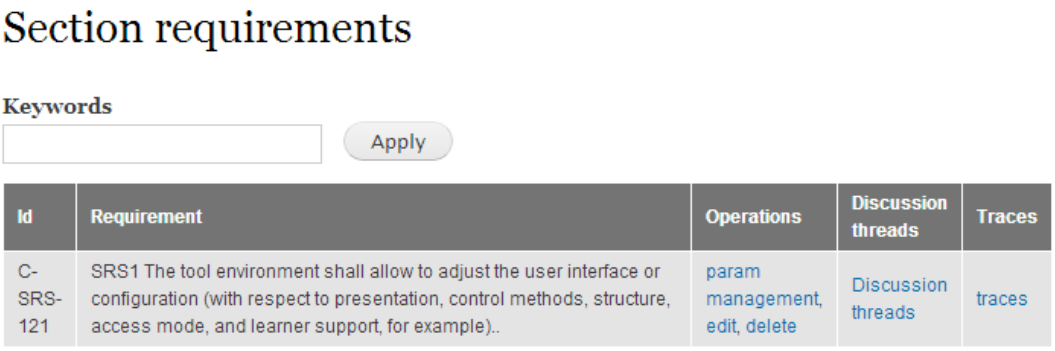


Siguiendo con el documento de ejemplo crearemos las secciones 3.5.6 Internationalization y 3.5.6.1 General Aspects de igual modo que ya hemos hecho. Navegamos por secciones y subsecciones hasta llegar a sección 3.5.6.1 y pulsamos sobre “add requirement”:

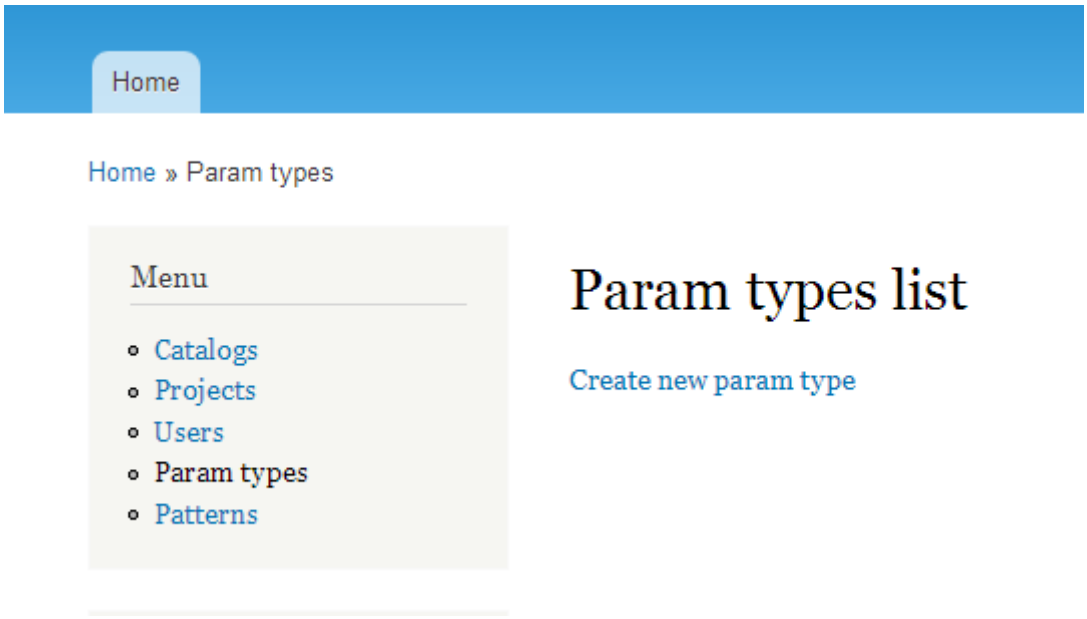
Añadimos un requisito a la sección:



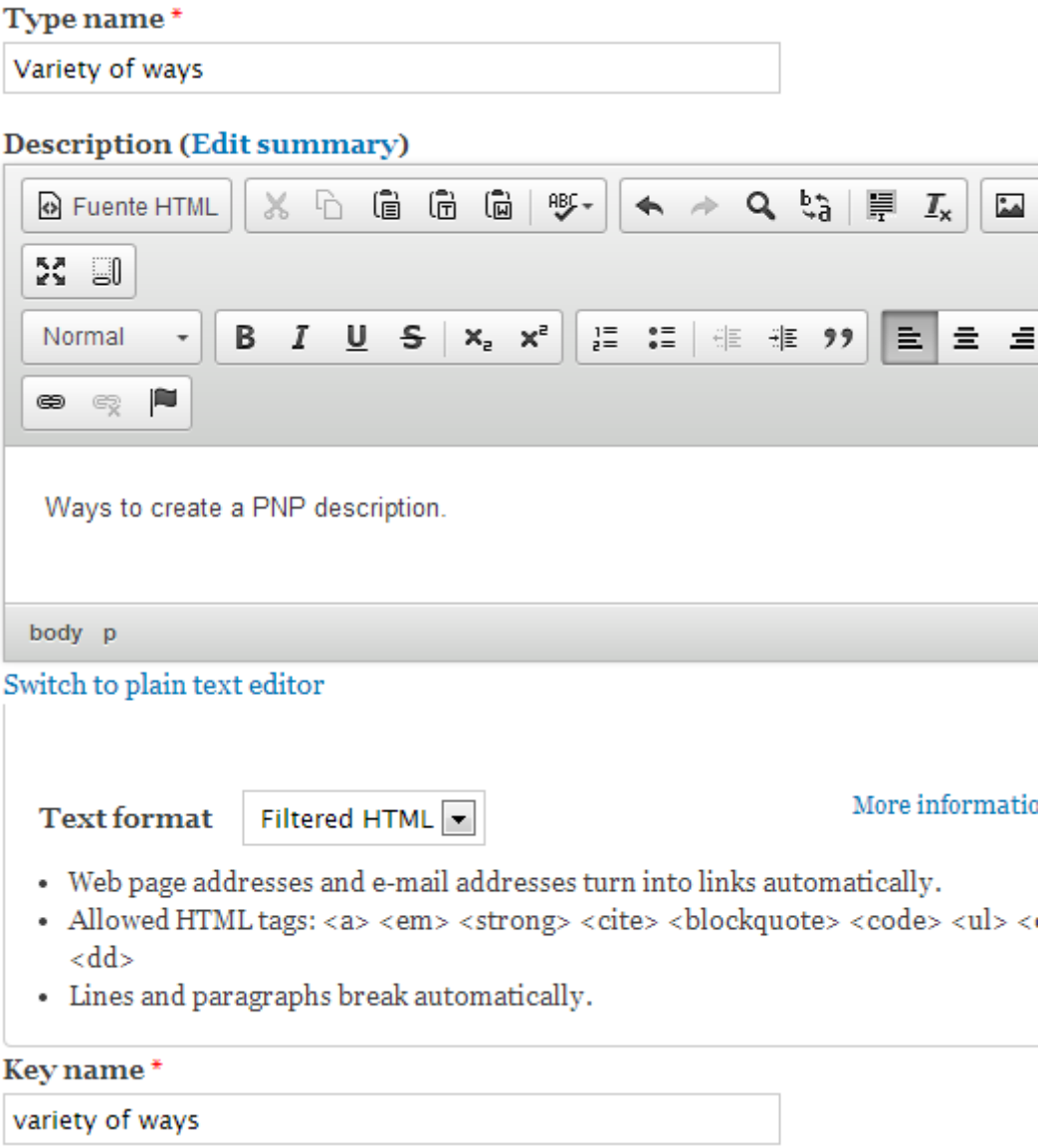
Indicar que el nombre no es el identificador de requisito que le asigna automáticamente PANTALASA-CMS, es sólo un nombre descriptivo. Si nos vamos a la sección se nos mostrará el requisito creado.



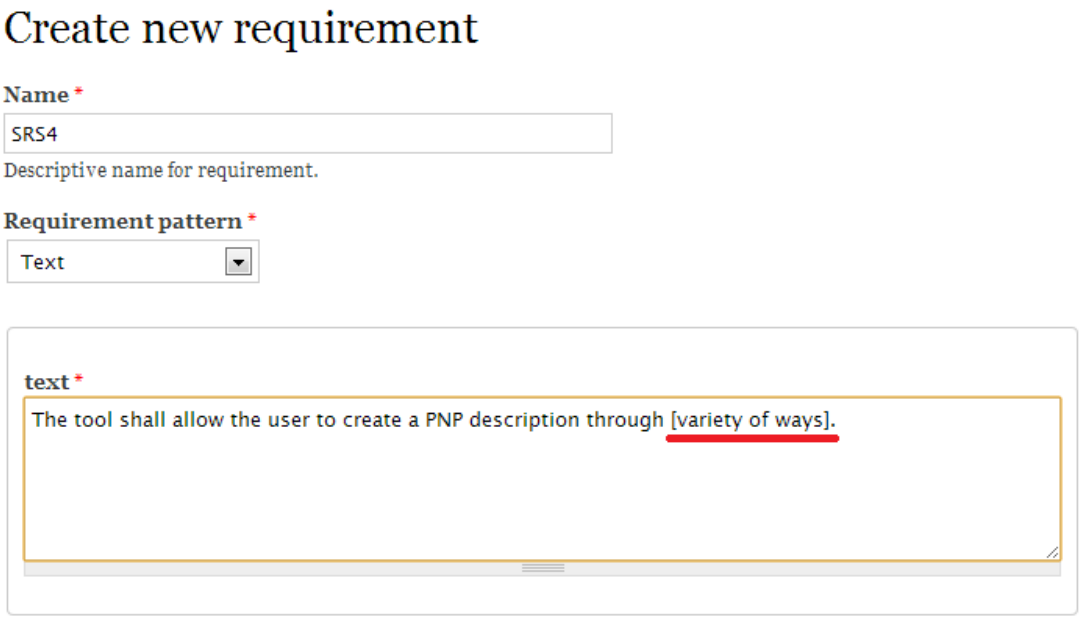
Vamos a añadir ahora un tipo de parámetro. Desde la página principal del responsable nos vamos a “Param types” y nos mostrará el listado de tipos de parámetros disponibles y una opción para añadir un nuevo tipo:



Pulsamos sobre “Create new param type” y creamos el nuevo tipo de parámetro:

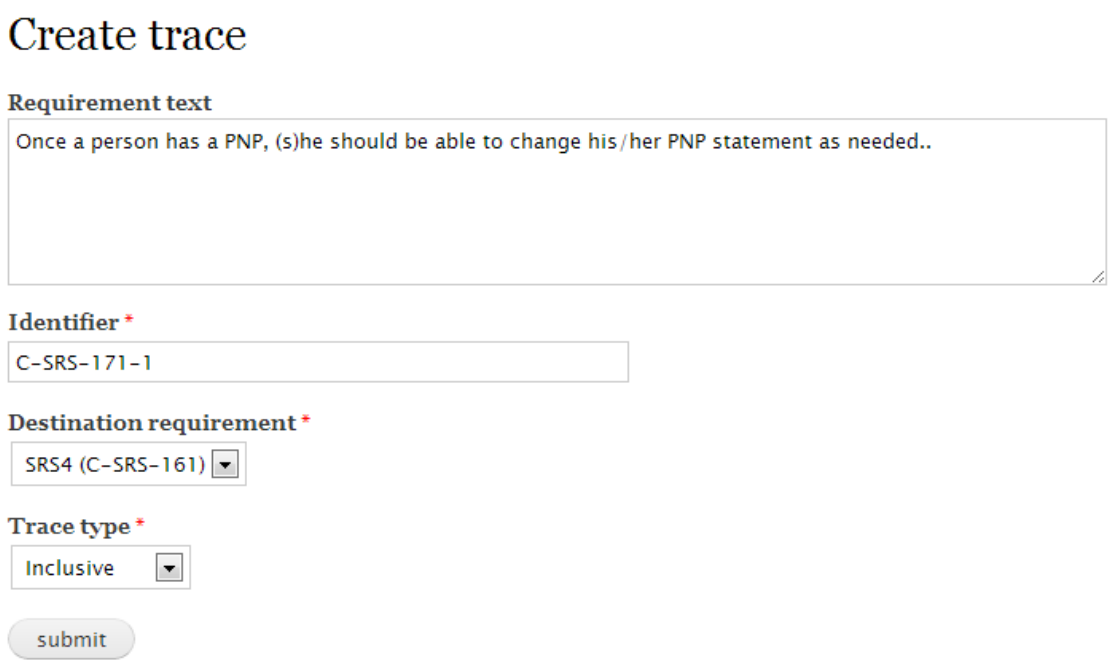


El “key name” es quien nos permitirá añadir parámetros a un requisito y poder instanciarlo poniéndolo entre corchetes “[ ]”. Vamos a verlo con un requisito:

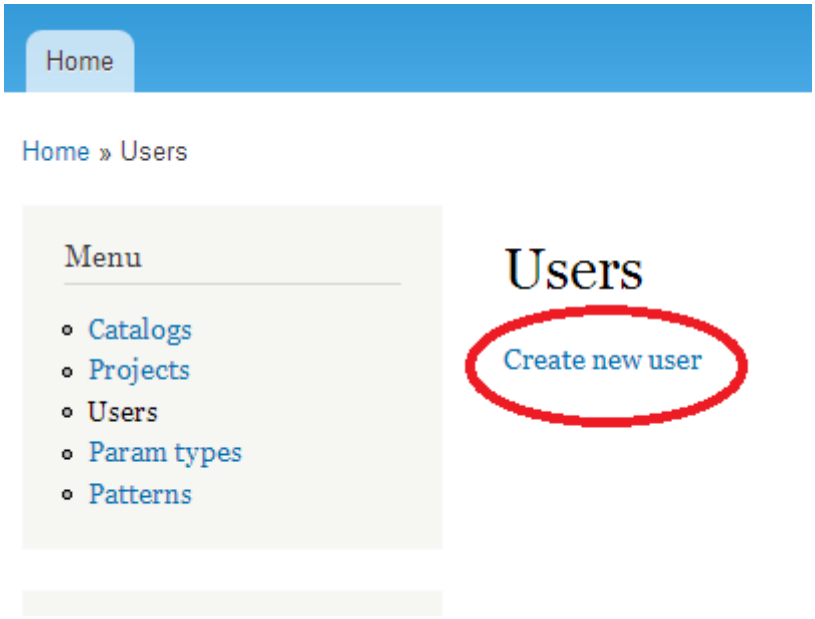


El nombre entre "[ ]" debe ser el "key name" del tipo de parámetro. Más adelante veremos cómo se instancia. Ahora vamos a añadir una traza entre requisitos. Vamos al requisito al que queremos añadir la traza, el requisito origen y pulsamos sobre "traces". Se nos mostrará un listado de las trazas que tiene ya y una opción para añadir una nueva traza:

En el formulario de creación se nos muestra el texto del requisito a añadir la traza y se nos da a elegir entre los requisitos del catálogo y el tipo de traza:



Vamos ahora a crear un nuevo usuario de tipo coordinador. Para ello nos vamos a "Users" y a "Create new user":



Rellenamos su información:

Create new user

Name *

usercoord

The unique user name.

Password

Password

. . . . . .

Password strength: **Fair**

Confirm password

. . . . . .

Passwords match: yes

To make your password stronger:

- Add uppercase letters
- Add numbers
- Add punctuation

Email *

user@pantalasa-cms.net

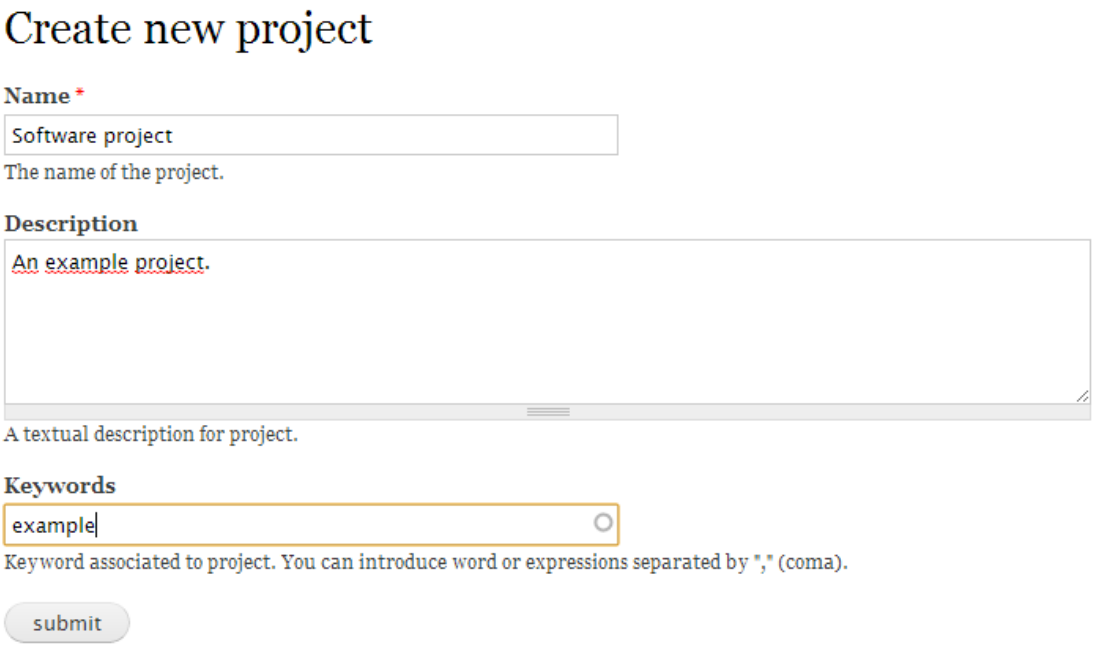
User types

coordinator

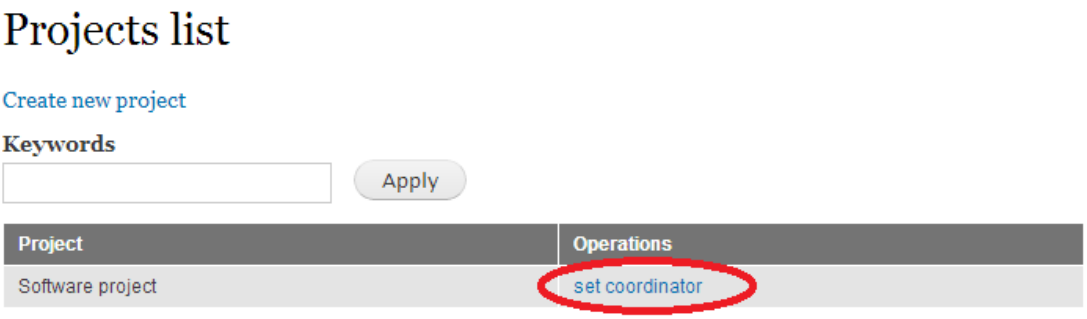
The user role.

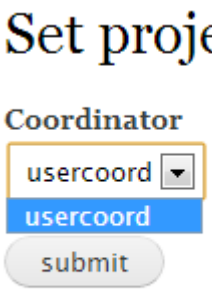
submit

Ahora creamos un proyecto y establecemos como coordinador de ese proyecto al usuario que acabamos de crear. Nos vamos a "Projects" y dentro a "Create new Project":



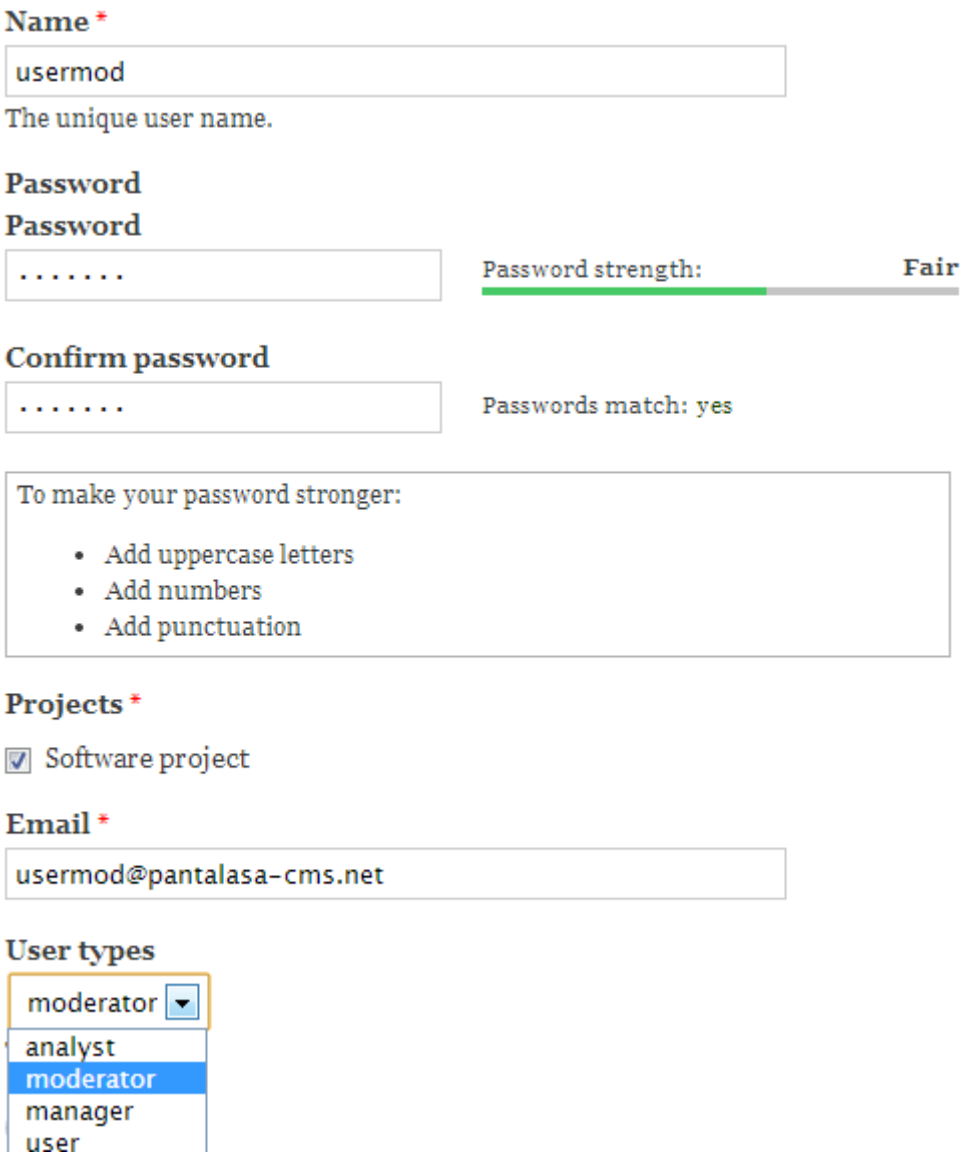
Ahora, en el listado de proyectos pulsamos sobre la opción "set coordinator" y seleccionamos un usuario:



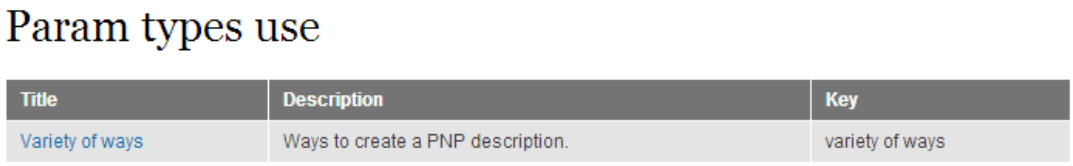


Ahora podemos entrar con el usuario coordinador y ver que es miembro del proyecto:

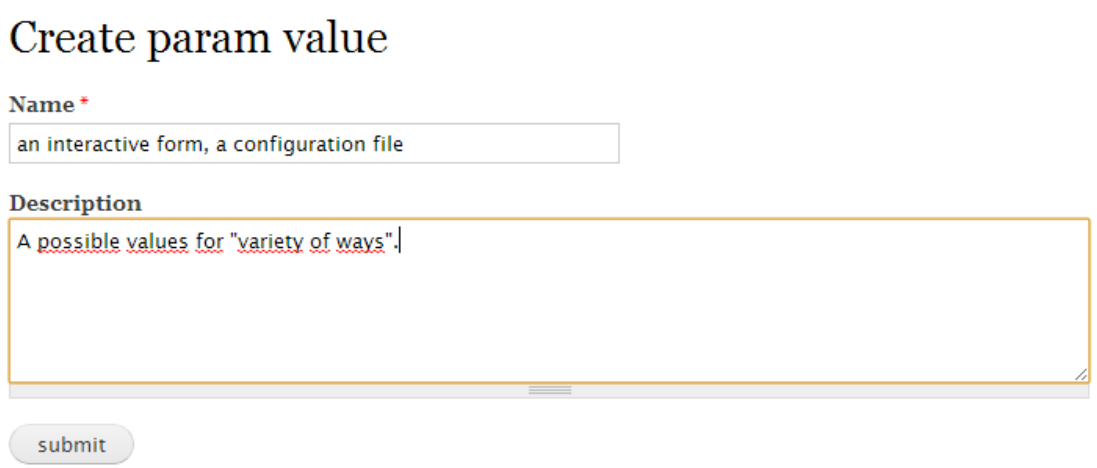
De la misma forma que antes crearemos un usuario moderador pero esta vez será el coordinador quien lo cree y lo asigne al proyecto:



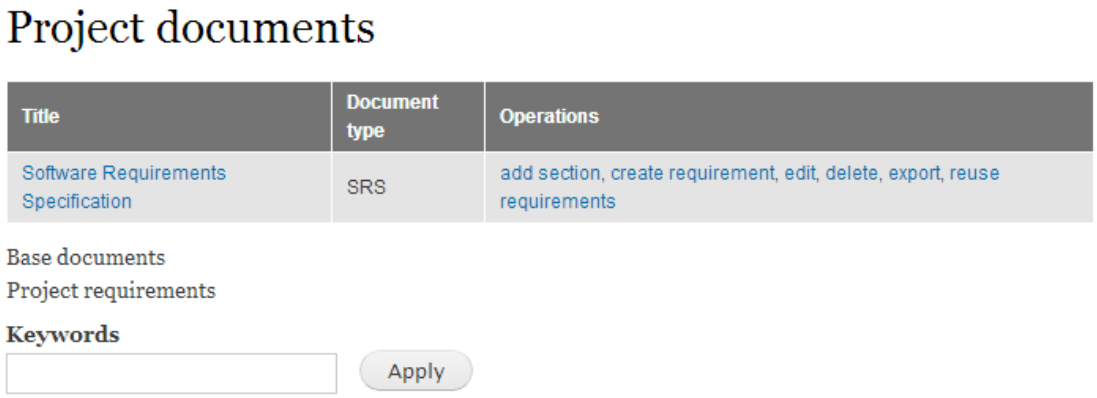
Entramos al sistema como el usuario moderador recién creado. Lo primero que haremos será crear un valor para el tipo de parámetro "variety of ways". Nos vamos a "Param types" y se nos mostrará un listado de tipos de parámetros:

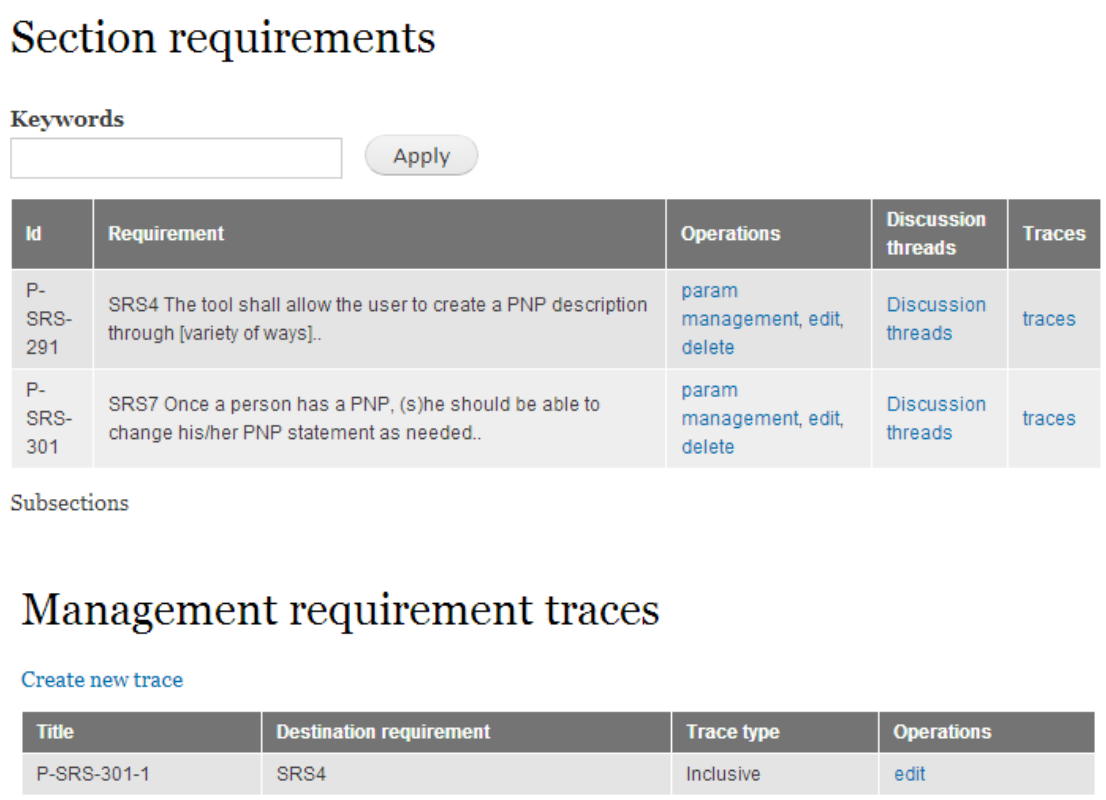


Pulsando sobre el nombre podemos ver los valores existentes y añadir un nuevo valor:

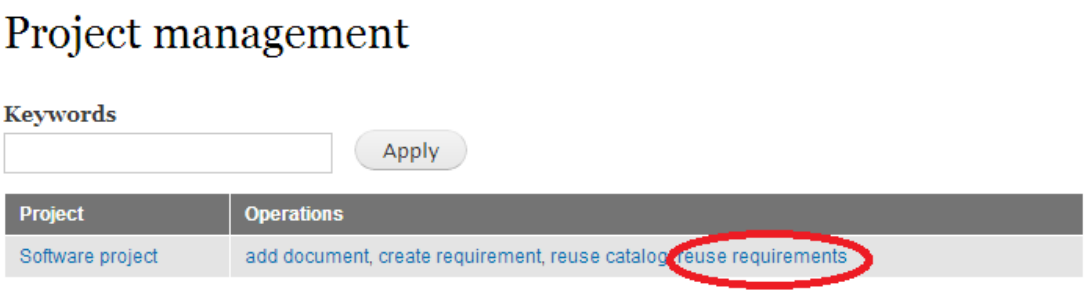


Vamos a reusar un catálogo completo. Nos vamos al proyecto y pulsamos sobre la opción "reuse catalog". Se nos mostrará un listado de catálogos disponibles. Pulsamos sobre el que queramos y se copiarán los documentos, secciones, requisitos y trazas. Abrimos el proyecto y vemos que se ha copiado:

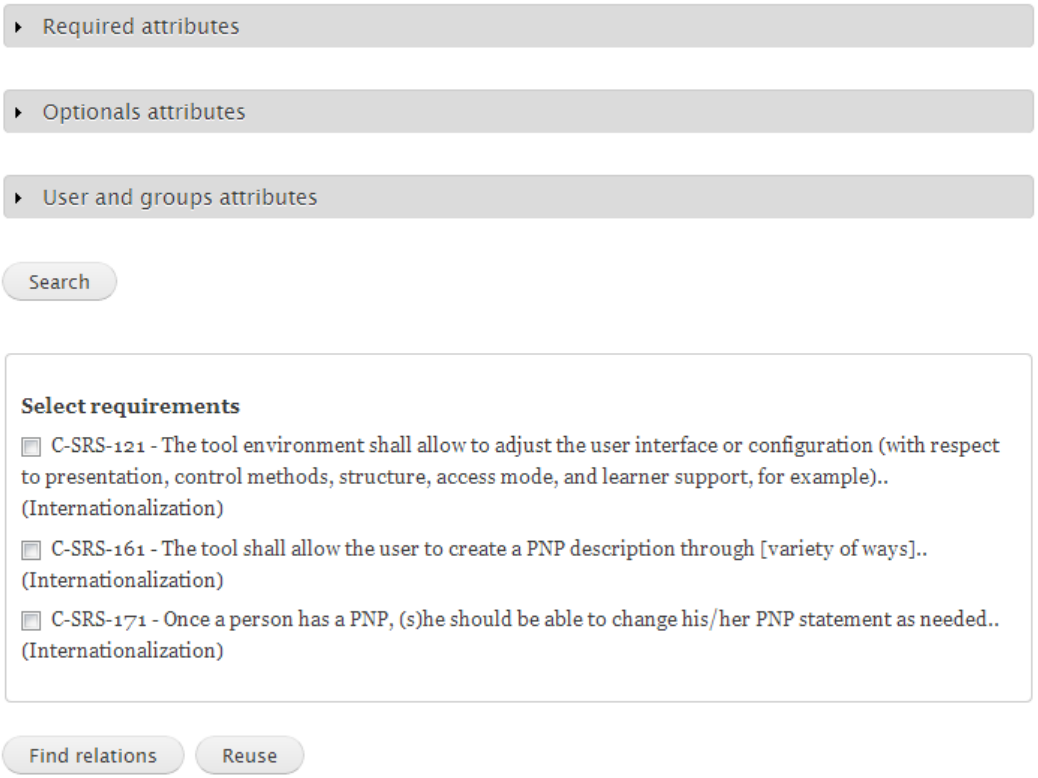




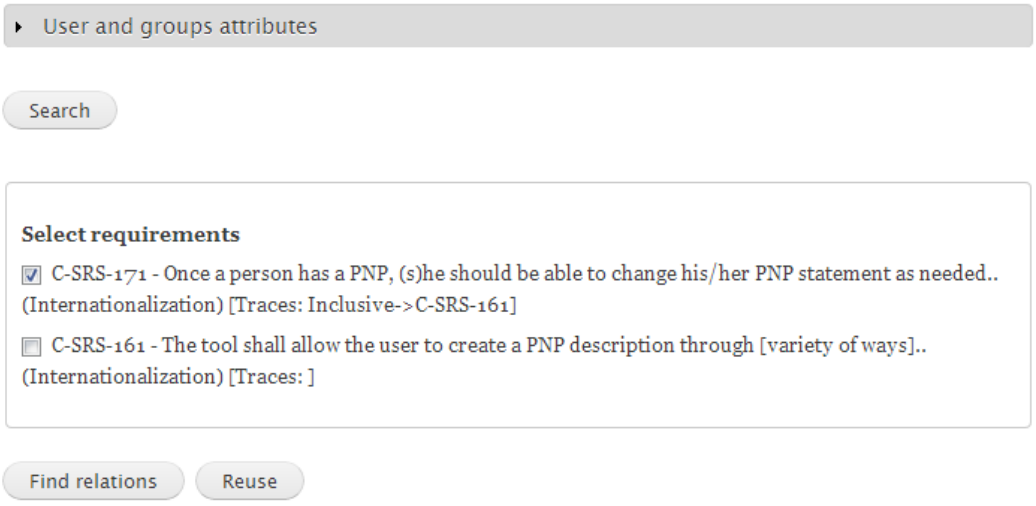
Ahora vamos a reusar un requisito del catálogo. Para ello nos vamos al lugar donde queremos reusarlo, en nuestro caso, en la raíz del proyecto. Pulsamos sobre “reuse requirements”:

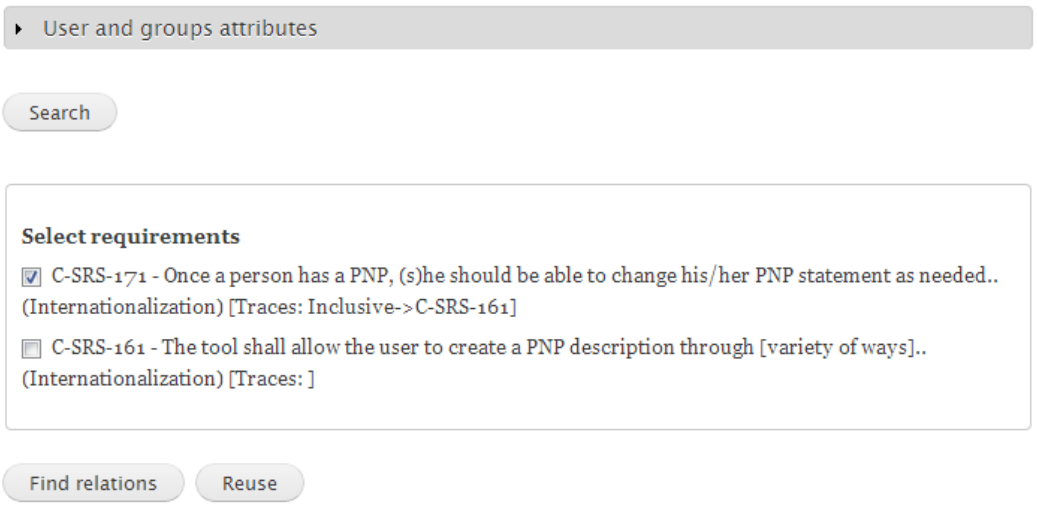


Se nos abre un buscador con multitud de filtros. En este caso no aplicamos ninguno para que nos muestre todos los requisitos de los catálogos del repositorio:

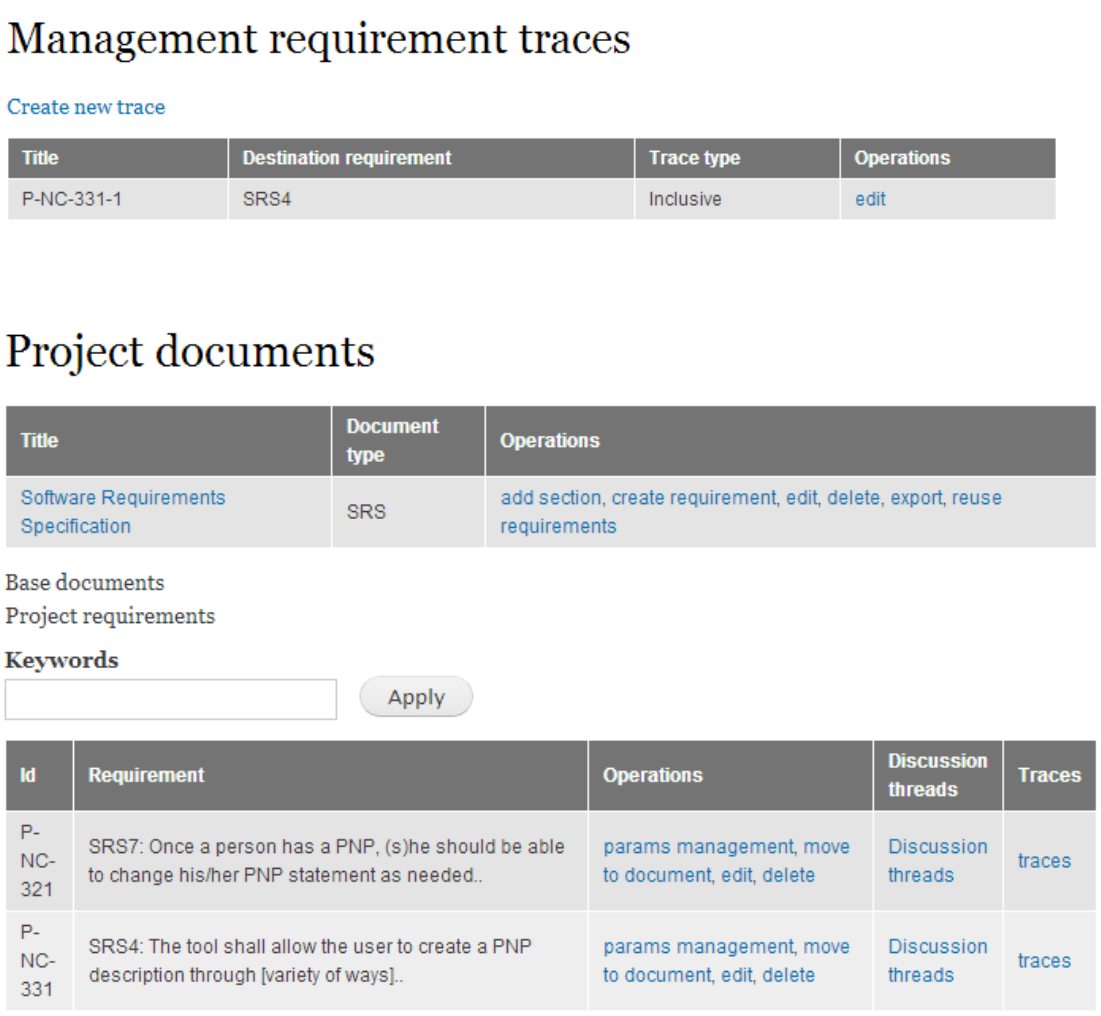


Seleccionamos el requisito C-SRS-161 y buscamos las relaciones que tiene con otros requisitos del mismo catálogo:





Vemos que el requisito C-STS-171 tiene una traza inclusiva con C-SRS-161 y no muestra el requisito por si queremos añadirlo. Lo seleccionamos y se copian los requisitos y la traza:



Por último vamos a ver cómo instanciar un parámetro. Para ello seleccionamos uno de los requisitos que tenga un tipo de parámetro y pulsamos sobre “params management”. Se nos abrirá un formulario con un select por cada parámetro:

6 Conclusiones y vías futuras

6.1 Conclusiones

Tras finalizar la realización de PANTALASA-CMS se ha lleado a una serie de conclusiones sobre el trabajo realizado y los objetivos logrados.

Esta versión de PANTALASA-CMS es una primera aproximación con aspectos básicos de soporte a PANGEA y aun quedan muchos aspectos a seguir tratando y mejorando en futuras versiones de la herramienta.

Para la implementación de la herramienta ha sido necesario un extenso periodo de formación en el API de Drupal para conocer y aprovechar todas las ventajas que ofrece este potente CMS. Como hemos dicho en el estudio de alternativas, Drupal es uno de los CMS más compliados de usar para la implementación de una herramienta de este tipo debido a la cantidad de código necesaria para implementar cierta funcionalidad. Sin embargo, una vez formado en ciertos aspectos, se aprecian las grandes ventajas que ofrece este CMS en cuanto a implementación de funcionalidad. Permite definir formularios muy personalizados y ampliar su potencial con tecnología AJAX, una gestión simplificada de manejar contenidos y abstraernos de su representación en base de datos, un control de grano fino sobre permisos de navegación y funcionalidades implementadas en el módulo y poder colaborar con otros módulos existentes en Drupal.

Uno de los aspectos en los que se ha tenido que invertir mucho tiempo y esfuerzo ha sido en el control de acceso y gestión de permisos. La gestión de permisos que ofrece Drupal es demasiado simple para la implementación de nuestra herramienta y ha sido necesario realizar toda esta gestión de forma manual, es decir, hemos tenido que implementar todo el código de control de permisos. Esta labor se ha hecho menos tediosa gracias al API de Drupal, ya que debido a que toda la funcionalidad implementada se ha realizado a través de URLs y en el hook_menu de Drupal puedes establecer una función que devuelve un valor booleano que indque si se tiene o no acceso. Esto también ha ayudado a mantener el código limpio y separado ya que cada funcionalidad se ha podido implementar en ficheros separados de unas pocas líneas.

Así como no es tan ardua la tarea de realizar el control sobre opciones de navegación y funcionalidad, no podemos decir lo mismo sobre el control de acceso a las vistas. Pese a la gran potencia que ofrece el módulo “Views” tenemos que decir que aun tiene muchos aspectos por mejorar y sin duda uno es el del control de permiso. El control que ofrece este módulo sobre usuarios y permisos es muy simple y no va más allá de controlar el autor de un contenido o el rol de un usuario. Esto nos ha facilitado la craeción de vistas ya que se han hecho vistas muy genéricas válidas para muchos roles de usuarios pero donde hemos tenido que ingeniárnoslas ha sido en el acceso a la vista, y de nuevo, gracias a la potencia de Drupal, hemos conseguido el resultado deseado. Justo antes de mostrar cada vista, hay una función del hook que se ejectua y recibe como parámetro la vista. Ahí es donde hemos realizo el control de qué vista se va a mostar y según la vista se ejecutará una u otra función para controlar el acceso. La ayuda que nos ofrece Drupal para realizar la implementación es una función que se ejecute donde se ejecuta nos deniega el acceso. Se ha utilizado para una vez

comprobado que nos e tiene acceso a una vista concreta se llama a esa función quedando una solución elegante con un código limpio y claro.

Otro aspecto que hemos de mencionar debido a su gran utilidad y a la potencia que ofrece es la gestión de formularios. Drupal proporciona una forma muy sencilla de crear formularios con variedad de componentes y todo ello creado desde un simple array PHP con una serie de valores y claves. Pero sin duda, lo que más ha sorprendido, es la facilidad de crear formularios usables aprovechando la potencia que nos ofrece AJAX. Con un simple array PHP con unos poco valores se ha podido hacer formularios dinámicos usables de forma sencilla.

Como aspecto negativo tenemos que hablar del tiempo de ejecuión. Debido a la cantidad de funciones y ficheros que se invocan para realizar una simple acción, por sencilla que sea, se hace notable la falta de rendimiento, sobre todo, en servidores compartidos. Decir, que el tiempo de instalación de un módulo puede llevar desde varios segundos hasta unos pocos minutos lo que hace que a la hora de la instalación sea necesario tener controlado el tiemout de nuestro servidor para que la instalación no se vea interrumpida.

En conclusión, Drupal es un potente CMS muy útil para sitios web sencillos como blogs o páginas estáticas donde en unas pocas horas puedes montar tu sitio web de forma visual y a golpe de click de ratón pero todo esto es prácticamente inútil cuando hemos de implementar una herramienta del estilo de PANTALASA-CMS. Hemos de recurrir a la craeción de un módulo y a la completa codificación y gestión de aspectos como el control de acceso y si además queremos que la instalación sea automatizada, hemos de codificar toda la parte de creación de contenidos. Pero sin duda, la funcionalidad y potencia que ofrece lo hace uno de los CMS de libre distrución más adecuados para este tipo productos software.

6.2 Vías futuras

Como aspectos a mejorar tenemos que hablar de las vistas de resultados y la navegabilidad entre las mismas. Como hemos mencionado, el módulo “Views”, pese a la funcionalidad y protencia que ofrece, tiene muchos aspectos que mejorar como es el control de acceso y los filtros de resultados. El código que ofrecen las vistas es muy cerrado en el sentido de las pocas modificaciones que permite realizar. Quizá, en futuras versiones del módulo se vea mejorado y permite una mejor navegabilidad y representación de resultados.

Si no fuera posibles podría optarse por la creación de un theme con plantillas propias y representar las consultas de contenidos mediante estas lo que permitiría un mayor que con las vistas.

Finalmente, otro aspecto que sin duda ha de tenerse en cuenta en un futuro, es la continua aparición y evolución de módulos para Drupal y las posibilidades que estos pueden ofrecer como extensiónes y mejoras a PANTALASA-CMS.

7 Referencias

[1] A. Muñoz Gallego and J. Rodríguez Lavado, Proyecto Informático: "Gestión de requisitos en el desarrollo global de software mediante SMW+," - 2013.

[2] A. A. López, Proyecto Informático: "SIRENgsd: Una propuesta para la reutilización de requisitos en el desarrollo global de software," ed.

[3] G. a. I. S. Kontoya, "Requirements Engineering. Processes and Techniques," ed, 1998.

[4] J. Nicolás, B. Moros, L. Joaquín, and A. Toval, "SIREN: Un Método Práctico de Ingeniería de Requisitos Basado en Reutilización," 2009.

[5] S. Diego and R. Fernando, "SopDGS: HERRAMIENTA CARE DE SOPORTE A PANGEA MEDIANTE REDES SOCIALES," ed, 2011.

[6] Compuarca.com, "CMS un pequeño listado de los mejores manejadores de contenido," www.compuarca.com%2Fcms-un-pequeno-listado-de-los-mejores-manejadores-de-contenido (ultimo acceso Septiembre 2013)

[7] D. W. Kumull, "Los 17 CMS más usados," http://dev.kumull.com%2Fcms%2Flos-17-cms-mas-usados.html (ultimo acceso Septiembre 2013)

[8] elwebmaster.com, "Lista de 15 CMS open source gratuitos," www.elwebmaster.com%2Farticulos%2Flista-de-15-cms-open-source-gratuitos (ultimo acceso Septiembre 2013)

[9] desarrolloweb.com, "Los mejores CMS Open Source," www.desarrolloweb.com/de_interes/mejores-cms-open-source-2174.html (ultimo acceso Septiembre 2013)

[10] cristalab.com, "Cuál CMS seleccionar??," http://foros.cristalab.com/cual-cms-seleccionar-t104563/ (ultimo acceso Septiembre 2013)

[11] Wordpress, "wordpress.org," (ultimo acceso Septiembre 2013)

[12] Joomla, "Joomla," www.joomla.org (ultimo acceso Septiembre 2013)

[13] M. CMS, "Magnolia CMS," www.magnolia-cms.com (ultimo acceso Septiembre 2013)

[14] wordpress.org, "Codex," http://codex.wordpress.org/Main_Page (ultimo acceso Septiembre 2013)

[15] ayudawordpress.com, "Drupal vs WordPress," http://ayudawordpress.com/drupal-vs-wordpress/ (ultimo acceso Septiembre 2013)

[16] webnova.com.ar, "Joomla: razones para usarlo en su sitio web," (ultimo acceso Septiembre 2013)

[17] chemalamiran.es, "¿WordPress o Joomla?," (ultimo acceso Septiembre 2013)

[18] http://isyourweb.com, "Comparativa Drupal, Joomla y Wordpress," (ultimo acceso Septiembre 2013)

[19] D. Hispano. Creación de módulos personalizados en Drupal. Available: http://drupal.org.es/node/12311 (ultimo acceso Septiembre 2013)

[20] sillygwailo. (2006, Writing module .info files (Drupal 7.x). Available: https://drupal.org/node/542202 (ultimo acceso Septiembre 2013)

[21] Drupal. Taxonomy module. Available: https://api.drupal.org/api/drupal/modules%21taxonomy%21taxonomy.module/7 (ultimo acceso Septiembre 2013)

[22] Drupal. function field_delete_field. Available: https://api.drupal.org/api/drupal/modules!field!field.crud.inc/function/field_delete_field/7 (ultimo acceso Septiembre 2013)

[23] Drupal. Node mocule. Available: https://api.drupal.org/api/drupal/modules%21node%21node.module/7 (ultimo acceso Septiembre 2013)

[24] Drupal. User module. Available: https://api.drupal.org/api/drupal/modules%21user%21user.module/7 (ultimo acceso Septiembre 2013)

[25] Drupal. Views 7. Available: https://drupal.org/project/views (ultimo acceso Septiembre 2013)
[26] Drupal, "Menu module,"
[27] Drupal. References module. Available: https://drupal.org/project/references (ultimo acceso Septiembre 2013)
[28] *C. T. LeeHunter, Daglees, chrisjlee*. (2013). *Organize content with taxonimies*.
[29] Drupal. node_type_save. Available: https://api.drupal.org/api/drupal/modules%21node%21node.module/function/node_type_save/7 (ultimo acceso Septiembre 2013)
[30] Drupal. function field_create_field. Available: https://api.drupal.org/api/drupal/modules!field!field.crud.inc/function/field_create_field/7 (ultimo acceso Septiembre 2013)
[31] Drupal, "Form API Reference," ed.

8 Anexos

8.1 Instalación

Para la instalación de PANTALASA-CMS necesitamos una instalación previa de la distribución de Drupal, concretamente, la versión 7.

Añadimos nuestro módulo al directorio “sites/all/modules/” de Drupal y colocamos todos los ficheros y directorios del módulo bajo “sites/all/modules/pantalasa_cms/”.

Antes de nada, tenemos que editar el fichero llamado “config.inc”. Este fichero contiene una función llamada “getDrupalPath” que devuelve la ruta absoluta al directorio de instalación de Drupal. Si tenemos un directorio virtual o alguna configuración especial de nuestro servidor, es conveniente editar esa función con la ruta absoluta al directorio de Drupal. Si tenemos una instalación normal, es decir, el directorio de instalación es mismo que el directorio físico, no tendermos que modificar nada.

También debemos añadir permisos de escritura al directorio “patterns” para poder gestionar los patrones de requisitos.

Ahora, nos vamos al listado de módulo de Drupal y vemos que aparece nuestro módulo. Comprobamos que están disponibles todos los módulos que necesita y procedemos con la instalación.

Si todo ha hido bien se nos debe mostrar una serie de mensajes informando de que se ha habilitado el módulo.

Ahora nos dirigimos a la página de inicio y ejecutamos el asistente. Consta de cuatro simples pasos donde sólo hay que ir avanzado y comprobando que los mensajes mostrados son correctos. Esto se hace así para evitar problemas de timeout con los servidores y ejecutar todas las acciones de forma gradual y evitar inconsistencias.

Ya está todo instalado pero falta realizar unos pequeños ajustes. Debido a las depencias entre vistas en el momento de la instalación, algunas referencias no es posible satisfacerlas automáticamente y ha de hacerse manualmente. En el siguiente listado se nombran las vistas que hay que modificar y cómo ha de hacerse:

(Sugerencia: Antes de revisar estos cambios es conveniente limpiar la caché de Drupal ya que al hacerlo resuelve las referencias. No obstante, es conveniente realizar este chequeo en las vistas ya que los filtros no son resueltos.)

Vista	Display	Sección	Ajuste
Project documents management	Ambos	Footer	First view area: base_documents - Display: block
			Second view area: requisitos_de_un_proyecto: - Display: block_1
		Filter	Document state: Modifiable

Vista	Display	Sección	Ajuste
Project documents (coordination)	Coordinación	Footer	First view area: base_documents - Display: block
		Filter	Document state: Modifiable
Catalog documents (management)	Gestión	Footer	First view area: requisitos_de_un_cat_logo - Display: block
	Query	Footer	First view area: requisitos_de_un_cat_logo - Display: block_2
Document sections	Secciones de un documento	Footer	First view area: requisitos_de_un_documento - Display: page
	Query	Footer	First view area: requisitos_de_un_documento - Display: page_1
Subsection requirements	Requisitos de sección	Footer	First view area: subsections - Display: block
	Query	Footer	First view area: subsections - Display: block_1
Section requirements	Requisitos de sección	Footer	First view area: subsections - Display: block
	Query	Footer	First view area: subsections - Display: block_1
Base documents	Base documents	Filter	Document state: Base
	Query	Filter	Documetn state: Base

Este es el proceso de instalación. Para acceder al sistema lo hacemos con el usuario responsable los siguientes datos:

- Usuario: responsable
- Password: responsable

Este usuario es el que crea los proyectos y les asigna los usuarios coordinadores que también crea.

8.2 XML Schema de un documento de requisitos

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns=http://www.w3.org/2001/XMLSchema
targetNamespace="http://www.example.org/document"
      xmlns:tns="http://www.example.org/document"
lementFormDefault="unqualified">
      <element name="document">
            <complexType>
                  <sequence>
                        <element name="previous_sections"
type="tns:previous_sections"
                              minOccurs="0"></element>
                        <element name="requirements"
type="tns:requirements" />
                        <element name="later_sections"
type="tns:later_sections"
                              minOccurs="0" />
                        <element name="sections" type="tns:sections" />
                  </sequence>
                  <attribute name="title" type="string" />
                  <attribute name="description" type="string" />
                  <attribute name="type">
                        <simpleType>
                              <restriction base="string">
                                    <enumeration value="IRS" />
                                    <enumeration value="SyRS" />
                                    <enumeration value="SRS" />
                                    <enumeration value="STR" />
                                    <enumeration value="SyTS" />
                              </restriction>
                        </simpleType>
                  </attribute>
                  <attribute name="state">
                        <simpleType>
                              <restriction base="string">
                                    <enumeration value="Modifiable" />
                                    <enumeration value="Listed" />
                                    <enumeration value="Base" />
                              </restriction>
                        </simpleType>
                  </attribute>
            </complexType>
```

```
	</element>

	<complexType name="previous_sections">
		<sequence>
			<element name="value" type="string" />
			<element name="format" type="string" />
		</sequence>
	</complexType>

	<complexType name="later_sections">
		<sequence>
			<element name="value" type="string" />
			<element name="format" type="string" />
		</sequence>
	</complexType>

	<complexType name="requirements">
		<sequence>
			<element name="requirement" type="tns:requirement"
minOccurs="0"
				maxOccurs="unbounded" />
		</sequence>
	</complexType>

	<complexType name="requirement">
		<sequence>
			<element name="traces" type="tns:traces" />
			<element name="params" type="tns:params" />
		</sequence>
		<attribute name="identifier" type="string" />
		<attribute name="title" type="string" />
		<attribute name="text" type="string" />
		<attribute name="risk">
			<simpleType>
				<restriction base="string">
					<enumeration value="High" />
					<enumeration value="Medium" />
					<enumeration value="Low" />
				</restriction>
			</simpleType>
```

```
		</attribute>
		<attribute name="criticity">
			<simpleType>
				<restriction base="string">
					<enumeration value="High" />
					<enumeration value="Medium" />
					<enumeration value="Low" />
				</restriction>
			</simpleType>
		</attribute>
		<attribute name="priority">
			<simpleType>
				<restriction base="string">
					<enumeration value="High" />
					<enumeration value="Medium" />
					<enumeration value="Low" />
				</restriction>
			</simpleType>
		</attribute>
		<attribute name="state">
			<simpleType>
				<restriction base="string">
					<enumeration value="Approved" />
					<enumeration value="Defined" />
					<enumeration value="Discarded" />
					<enumeration value="Implemented" />
					<enumeration value="Modeling analysis" />
					<enumeration value="Modeling design" />
					<enumeration value="Still to be" />
					<enumeration value="Pending discussion"
/>
					<enumeration value="Pending Review" />
					<enumeration value="Verified" />
				</restriction>
			</simpleType>
		</attribute>
		<attribute name="fulfillment">
			<simpleType>
				<restriction base="string">
					<enumeration value="Mandatory" />
					<enumeration value="Optional" />
```

```
                        <enumeration value="Advisable" />
                    </restriction>
                </simpleType>
            </attribute>
            <attribute name="verification">
                <simpleType>
                    <restriction base="string">
                        <enumeration value="Analysis" />
                        <enumeration value="Demostration" />
                        <enumeration value="Inspection" />
                        <enumeration value="Test" />
                    </restriction>
                </simpleType>
            </attribute>
            <attribute name="motivation" type="string" use="optional"
/>
            <attribute name="source" type="string" use="optional" />
            <attribute name="proposed-by" type="string" use="optional"
/>
            <attribute name="source-workgroup" type="string"
use="optional" />
            <attribute name="destination-workgroup" type="string"
use="optional" />
            <attribute name="analyst" type="string" use="optional" />
      </complexType>

      <complexType name="sections">
            <sequence>
                <element name="section" type="tns:section"
minOccurs="0"
                    maxOccurs="unbounded" />
            </sequence>
      </complexType>

      <complexType name="section">
            <sequence>
                <element name="requirements" type="tns:requirements"
/>
            </sequence>
            <attribute name="title" type="string" />
            <attribute name="description" type="string" use="optional"
/>
```

```
		<attribute name="number" type="integer" use="optional" />
	</complexType>

	<complexType name="traces">
		<sequence>
			<element name="trace" type="tns:trace" minOccurs="0"
				maxOccurs="unbounded" />
		</sequence>
	</complexType>

	<complexType name="trace">
		<attribute name="requirement" type="string" />
		<attribute name="type">
			<simpleType>
				<restriction base="string">
					<enumeration value="Exclusive" />
					<enumeration value="Inclusive" />
					<enumeration value="Father_soon" />
					<enumeration value="Related" />
				</restriction>
			</simpleType>
		</attribute>
	</complexType>

	<complexType name="params">
		<sequence>
			<element name="param" type="tns:param" minOccurs="0"
				maxOccurs="unbounded" />
		</sequence>
	</complexType>

	<complexType name="param">
		<attribute name="key" type="string" />
		<attribute name="value" type="string" />
	</complexType>

</schema>
```

8.3 XML Schema de un patrón

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.example.org/requirementPattern"
      xmlns:tns="http://www.example.org/requirementPattern"
      elementFormDefault="qualified">
      <element name="requirement-pattern">
            <complexType>
                  <sequence>
                        <element name="pattern-part" type="tns:pattern-
part"
                              minOccurs="0" maxOccurs="unbounded" />
                  </sequence>
                  <attribute name="name" type="string" use="required"
/>
                  <attribute name="description" type="string" />
            </complexType>
      </element>

      <complexType name="pattern-part">
            <choice>
                  <element name="keyword" type="tns:keyword" />
                  <element name="expression-name" type="tns:expression"
/>
                  <element name="constant" type="tns:constant" />
            </choice>
      </complexType>

      <complexType name="keyword">
            <attribute name="key-name" type="string" use="required" />
      </complexType>

      <complexType name="expression">
            <attribute name="expression-name" type="string"
use="required" />
            <attribute name="required-expression" type="boolean"
use="optional"
                  default="false" />
                  <attribute name="field-type" default="text">
                        <simpleType>
                              <restriction base="string">
                                    <enumeration value="text" />
```

```
                                    <enumeration value="long_text" />
                                </restriction>
                            </simpleType>
                        </attribute>
        </complexType>

        <complexType name="constant">
              <attribute name="value" type="string" use="required" />
        </complexType>

</schema>
```