Grado en Ingeniería Informática
2016-2017

Área de Gestión del Conocimiento

Trabajo Fin de Grado

"*FLOSS NET.*

Free/Libre and Open Source Software Network"

Enrique Hidalgo Ayllón

Tutor/es

Javier Martí Pintanel

En Barcelona, 19 de junio de 2017

Agradecimientos

A mis padres y hermanos, por depositar su confianza y ofrecerme el apoyo constante y necesario durante toda mi trayectoria personal, laboral y educativa.

A Javier Martí, tutor del proyecto y del Grado en Ingeniería Informática, por ofrecerme su ayuda y colaboración a lo largo de mi etapa educativa, especialmente durante la ejecución de este proyecto.

Agradecimiento especial a Mireia Castro, que ha compartido gran parte de esta aventura conmigo ofreciéndome su apoyo y cariño en todos los momentos, especialmente en las etapas y momentos más duros.

Resumen

El trabajo de fin de grado *"FLOSS NET. Free/Libre and Open Source Software Network"* es el resultado de aplicación del proceso de ejecución un proyecto de Ingeniería Software, a partir de un problema a una necesidad detectada. En este marco, se incluye el contexto del proyecto, con la creación de un portal web que permita establecer un punto de partida a colaboraciones de proyectos basados en tecnologías FLOSS, para ofertantes de servicios y demandantes con proyectos basados en estas.

Para acometer el proyecto se plantea como solución el uso de Liferay Portal, con el desarrollo de las funcionalidades a modo de *Portlets*. El proyecto plantea una solución al problema mediante la elaboración del análisis de la solución, diseño y desarrollo de la misma, empleando recursos y estructuras que o bien la propia plataforma ofrece, o las plataformas para su desarrollo facilitan.

Palabras clave: Liferay, Portlet, Ingeniería del Software, CMS, FLOSS

Abstract

The Project "FLOSS NET. Free/Libre and Open Source Software Network" is the result of applying the execution of Software Engineer Process, from the problem of a detected need. In this context, it's included the current Project, that introduces the creation of a web portal that allow to establish a starting point for Project collaborations based on FLOSS technologies, for offerers and applicants of these technologies.

In order to carry out the Project it is proposed as solution the use of Liferay Portal, developing functionalities as Portlets. The Project presents the problem solution by the analysis, design and development of the portal, using resources and structures of the main platform or development frameworks.

Keywords: Liferay, portlet, Software Engineering, CMS, FLOSS

Índice de contenidos

Capítulo 1. Planteamiento e Introducción

1.1 Contexto

Con el auge de las tecnologías de la información y las comunicaciones (TIC), quedó al alcance de las empresas la posibilidad de control y gestión sobre sus propios recursos de manera significativa. Mediante la implantación de herramientas desarrolladas, se consiguieron grandes avances en optimización de recursos y niveles de operatividad impensables.

La gestión de información y proceso mediante herramientas Tics aportan valor añadido a las actividades empresariales y de gestión empresarial y permiten obtener numerosas ventajas competitivas, permanecer en el mercado y focalizar los esfuerzos en el núcleo de negocio de la empresa y no en las otras actividades.

Las Tics pertenecen a una clase emergente de aquellas tecnologías que referencia al uso de medios informáticos para almacenar, procesar/gestionar y difundir cualquier tipo de información entre unidades o departamentos de una misma empresa, e incluso con clientes o usuarios externos.

No obstante, esta facilidad de implantación esconde un déficit de conocimientos técnicos que las organizaciones deben afrontar mediante la contratación de recursos propios especialistas en la materia o empresas y/o trabajadores externos del sector.

Por ello, es indispensable realizar un uso eficiente de estas tecnologías, encontrando los procedimientos adecuados y adaptándolos a cada necesidad para la obtención de ventajas competitivas ya que, a pesar de presentar herramientas potentes para el negocio de una empresa, la implantación de un sistema por sí solo no implica beneficios directos. Para obtener dichos beneficios, es necesario llevar a cabo un planteamiento estratégico tomando en cuenta diversos factores, como por ejemplo las necesidades actuales y futuras de la empresa.

Por otro lado, en aquellos casos en que la implantación se realice de manera correcta las TIC se convertirán en elementos esenciales de la empresa en su mejora de productividad, calidad, control y comunicación.

En el caso que atañe **a este proyecto**, **se tratarán los sistemas de tipología FLOSS (*Free/Libre and Open Source Software* -Software libre y de código abierto)**. Los sistemas FLOSS, a diferencia del resto, se caracterizan por permitir a los usuarios su uso y a los profesionales el acceso al código fuente de los mismos. En este sentido, los profesionales

pueden acceder y modificar el código fuente de un sistema para adaptarlo a determinadas necesidades de casos concretos.

1.2 Problemática a resolver

Estas herramientas, a pesar de encontrarse diseñadas y desarrolladas para cumplir con los requisitos y necesidades que puedan aparecer en el mayor número posible de empresas, a menudo no se adaptan las necesidades concretas de cada empresa ya que están confeccionadas como módulos desarrollados de manera genérica que necesitan ser parametrizados o recodificados a medida para determinadas funcionalidades específicas de cada cliente en cuestión.

Se trata de sistemas que están extendiendo cada vez más su uso y el cual cada vez más empresas apuestan por implantar. Sin embargo, esta tipología de software cuenta con una importante desventaja. Los sistemas de software privados, por ejemplo, los sistemas ERP SAP y Navision, disponen de un mecanismo claro de atención técnica para las implantaciones, directo por parte del fabricante o a través de empresas colaboradoras. En cambio, los sistemas FLOSS no tienen un referente técnico claro para la empresa donde se ha implementado un sistema de este paradigma. Sí que todos estos sistemas cuentan con comunidades de usuarios, pero es referente a usuarios implantadores o profesionales informáticos, no a usuarios del propio sistema donde se ha realizado una implantación. Esta situación se produce en la gran mayoría de sistemas FLOSS y dificulta en gran medida el contacto y garantías para el usuario final, que es la empresa que está valorando la implantación de un sistema informático con este paradigma o tiene dificultades con una implantación ya realizada, cuando no dispone de personal propio (o en su defecto, socio tecnológico) especializado en estos sistemas informáticos.

En este sentido, para la implantación de un proyecto basado en sistemas FLOSS se recorren distintas fases de análisis de los procesos y datos de la Compañía, diseño a medida de los distintos diagramas de flujo que permitan modelar estos procesos, adaptación y parametrización de las soluciones desarrolladas de manera genérica, pruebas y validación de la solución, etc.

Es por este motivo, que las organizaciones emplean grandes cantidades de recursos ya sea de personal interno y/o colaboraciones externas cuando emprenden proyectos amparados en esta tipología de sistemas.

Del contexto planteado en el punto anterior, se detecta una variable limitante significativa, el hecho de ser necesaria la disposición de grandes recursos económicos o de personal interno por parte de las empresas para acometer estos proyectos.

Este aspecto no sería una variable limitante en un entorno donde prevalecieran las empresas con grandes estructuras y recursos para ejecutar e implementar proyectos de estas características. No obstante, si se observa el tejido empresarial nacional (véase el detalle en la siguiente ilustración) es posible determinar que el requisito indispensable de disponer de grandes recursos no se da en la mayoría de casos, con casi un 99,3% de empresas compuestas por **micro-empresas y pequeñas empresas**.

	Micro Sin asalariados *	Micro 1-9	Pequeñas 10-49	Medianas 50-249	PYME 0-249	Grandes 250 y más	Total
ESPAÑA	1.670.329	1.314.398	107.784	18.011	3.110.522	3.839	3.114.361
%	53,6	42,2	3,5	0,6	99,9	0,1	100
UE-28 %	92,4		6,4	1,0	99,8	0,2	100

Fuente: INE, DIRCE 2014 (datos a 1 de enero de 2014), y Comisión Europea, "ANNUAL REPORT ON EUROPEAN SMES 2013/2014" Estimaciones para 2013.
*Corresponde en su mayoría a personas físicas, ver tabla 7.

Ilustración 1 Distribución de empresas en el tejido empresarial de España. Año: 2014. Fuente: Instituto Nacional de Estadística (INE).

Las micro-empresas y pequeñas empresas se encuentran con obstáculos significativos a la hora de emprender la implantación de soluciones de SI ya que no disponen de recursos internos que den soporte y ejecuten las actividades derivadas de la ejecución del proyecto.

Es por ello que **se ven obligadas a emplear recursos externos para acometer estos proyectos**. Sin embargo, aunque para los sistemas software propietario es sencillo contactar con el desarrollador del sistema o *partners* de este para dar soporte en cuanto a deficiencias de funcionamiento u optimización del sistema a la casuística del cliente, en el caso del software libre **se generan grandes dificultades para establecer esta comunicación y encontrar al profesional o empresa apropiada, lo cual genera una confusión importante en la organización que quiere implantar o tiene dificultades con una implantación de este tipo de sistemas informáticos**.

1.3 Solución

1.3.1 Solución Planteada

En este marco nace la idea del proyecto, con el objetivo **generar un punto de encuentro entre microempresas y pequeñas empresas que deseen implantar sistemas FLOSS, por una parte, y por otra, los profesionales y empresas dedicadas a la implantación y mantenimiento de este tipo de sistemas**.

De este modo, se pretende facilitar a los usuarios una plataforma en la que ambas partes interesadas puedan acceder, de manera que éstas se puedan favorecer de un nuevo canal de contacto a través del posible desarrollo real del proyecto amparado en este TFG.

1.3.2 Alcance y Objetivos

Destacar que la solución se enfocará a **autónomos, micro-empresas y pequeñas empresas** interesadas, tanto como profesionales como usuarios de los sistemas FLOSS, puesto que las medianas y grandes empresas ya cuentan con recursos para llevar a cabo las implantaciones de los SI.

El objetivo principal del presente proyecto radica en el desarrollo de una plataforma web de contacto entre proveedores y solicitantes (micro-empresas y pequeñas empresas) de implantaciones de las plataformas, sistemas y soluciones existentes para el mundo de la empresa bajo el paradigma FLOSS, como solución al problema planteado en el apartado anterior.

El alcance de la plataforma no será meramente establecer el contacto entre los interesados, si no establecerá un sistema de reputación tanto para proveedores como para solicitantes de los servicios prestados, generando una base de conocimiento de valor añadido para ambos colectivos del portal.

El sistema implementará un flujo de información y contacto que permitirá tanto a solicitantes, como proveedores, disponer del primer contacto necesario para establecer la colaboración en una implantación de un sistema FLOSS.

En este sentido, los **solicitantes** podrán registrarse en el sistema y acceder a un listado de proveedores expertos para cada sistema FLOSS registrado en el portal para poder contactar con proveedores que le ofrezcan la implantación a medida.

Los solicitantes podrán realizar el contacto en base a las valoraciones y comentarios aportados por otros solicitantes del portal que hayan colaborado previamente con los proveedores.

Por otro lado, los **proveedores** podrán registrarse en el sistema y encontrar un nicho de solicitantes potenciales interesados en sus servicios, así como poder valorar y comentar también a los solicitantes para enriquecer la información de contacto del portal.

Este registro será necesario tanto por parte de los clientes como para proveedores para validar las condiciones legales asociadas al inicio de la colaboración.

Para la ejecución del proyecto, se ha seguido la metodología aprendida en la asignatura Gestión de Proyectos y la conveniente específica de cada materia según los conocimientos adquiridos a lo largo del plan de estudios.

1.4 Motivación y Conocimientos previos

La principal motivación que me ha llevado a plantear el presente proyecto es la tendencia actual de las empresas y clientes a establecer comunicación e intercambiar información vía redes sociales y la dificultad que afrontan las pequeñas y micro-empresas a la hora de implantar sistemas tecnológicos debido a la insuficiencia de sus recursos para acometer los proyectos.

En este sentido se plantea una gran opción con este proyecto en establecer un sistema informático de gestión de conocimiento que registre altas de usuarios que implemente un portal donde tanto proveedores como profesionales de soluciones OpenSource, y pequeñas y micro-empresas puedan acceder. Se desea además integrar en la solución técnicas y funcionalidades propias de un sistema de gamificación para que el contacto sea lo más beneficioso a ambas partes y dotar de valor añadido a la solución, haciendo más atractivo el uso del sistema.

En relación al conocimiento y experiencia previa en el entorno de trabajo, Actualmente trabajo dentro de una empresa de consultoría buscando financiación y subvenciones a

empresas cliente para acometer proyectos de I+D+i en el ámbito de las TIC. Las pequeñas y micro-empresas no disponen de los recursos necesarios para contratar colaboraciones externas para la implantación de software privativos a medida y las ayudas y subvenciones a nivel nacional y europeo están enfocadas a empresas que dispongan de una gran solvencia financiera.

Mediante la plataforma planteada, se pretende generar una herramienta que permita acceder a las empresas a la implantación de un sistema informático de software libre propio por empresas colaboradoras especialistas en estos, a un coste reducido.

1.5 Metodología

Para el desarrollo del presente proyecto se ha contado con el apoyo del tutor/director del Trabajo de Fin de Grado, Javier Martí Pintanel, que ha colaborado asesorando a lo largo de la ejecución del mismo desde la fase de planteamiento del proyecto, hasta el desarrollo del sistema prototipo planteado.

La metodología empleada en la ejecución del proyecto en referencia a la planificación y seguimiento del mismo, corresponde con la aprendida en la asignatura “Gestión de Proyectos” del Grado en Ingeniería Informática de la UOC.

De manera inicial, se planteó el proyecto y el alcance del mismo, así como se elaboró la planificación y recursos necesarios para llevarlo a cabo garantizando el éxito. Una vez definido el alcance y la planificación, se procedió al análisis de la problemática e investigación e identificación sobre posibles tecnologías que permitieran su implementación, a partir de distintos recursos bibliográficos.

Dado que la necesidad de definir, diseñar y desarrollar una plataforma web desde cero y totalmente funcional requiere una dedicación de recursos mayor a la disponible en el marco del Trabajo de Fin de Grado, la estrategia seguida para abordar la solución ha sido emplear plataformas de desarrollo, bajo el paradigma de Código Libre, que permitan el uso de componentes o funcionalidades ya desarrolladas por comunidades de programadores, simplificando la implementación de un prototipo inicial funcional.

Asimismo, se procedió a profundizar en la definición de la solución llevando a cabo la elaboración y análisis de requisitos. Posteriormente, se procedió a la elaboración del diseño del sistema y a su posterior codificación.

Finalmente, remarcar que, a lo largo de la ejecución del proyecto, se han elaborado un conjunto de entregables definidos en la planificación, anexados al final de esta memoria como informes de seguimiento (véase el Anexo 1 Informes de Seguimiento), indicando el avance y estado de ejecución de las distintas fases del proyecto.

1.6 Planificación

1.6.1 Introducción

Mediante las descripciones aportadas en este apartado se pretende ofrecer una perspectiva de los principales puntos en cuanto a la planificación del proyecto. Para ello se definen los principales hitos del proyecto, el desglose de actividades y tareas con los recursos necesarios para su ejecución, así como el análisis de riesgo de los principales problemas que podrían surgir durante la ejecución del mismo.

1.6.2 Hitos

El proyecto en cuestión se desarrolla dentro de la asignatura Trabajo de Fin de Grado del plan docente del Grado en Ingeniería Informática. En este sentido, el plan docente de la asignatura ha marcado un conjunto de hitos mediante los cuales será necesario aportar entregables ya definidos para el seguimiento y evaluación del proyecto.

En la siguiente tabla se puede observar el detalle de los hitos marcados en el calendario del TFG:

FECHA	HITO	Documentación/Descripción
15/03/2017	PAC1	Planteamiento+ Planificación + Informe de Seguimiento
03/04/2017	Identificación Tecnológica	Se seleccionan las tecnologías a aplicar en el proyecto
12/04/2017	PAC2	Informe de Seguimiento+Documentación (análisis y diseño)
21/05/2017	Obtención del producto	Se obtiene un prototipo funcional del proyecto
31/05/2017	PAC3	Informe de seguimiento+Documentación (diseño, implementación, pruebas y plan de implantación)
19/06/2017	Memoria, producto, presentación virtual y autoinforme	Memoria TFG+Informes de Seguimiento+Producto+Presentación virtual+Informe de Autoevaluación
30/06/2017	Publicación	Publicación en repositorio O2

Ilustración 2 Hitos del proyecto.

1.6.3 EDTs y tareas

Para la ejecución del proyecto del TFG se ha estimado una carga equivalente a 300 horas, correspondiente a 12 créditos ECTS. De esta carga, se estima que un 20% corresponderán a tareas estrictamente académicas de la asignatura TFG, por lo que el total de horas dedicadas íntegramente al desarrollo del producto contemplado en este TFG se estima en **240 horas**.

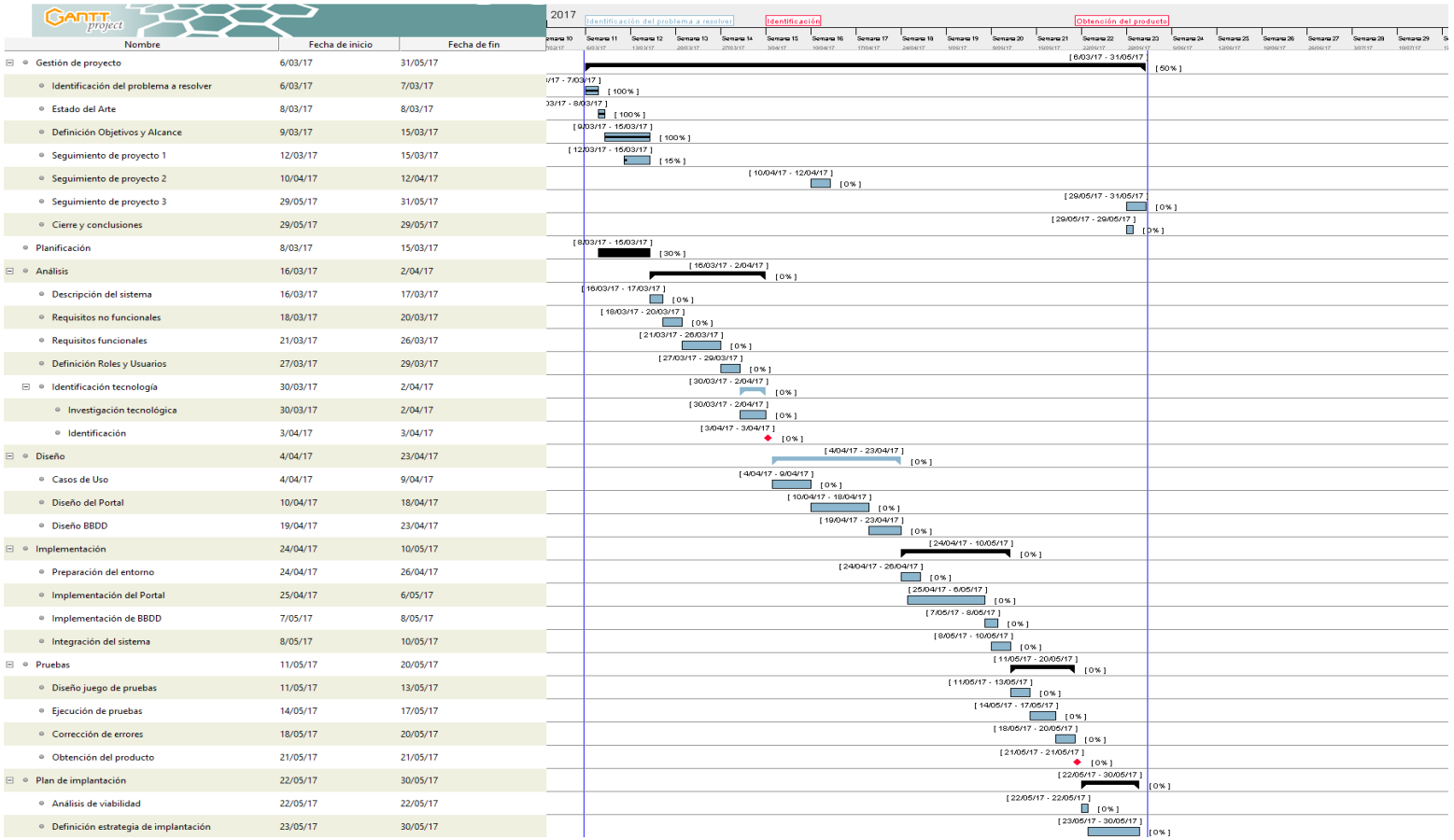
A partir de la carga estimada, se ha elaborado la siguiente planificación mediante Estructuras de Descomposición de Trabajo (EDTs), que tendrán una ejecución secuencial, siguiendo la metodología definida por la asignatura Gestión de Proyectos del plan docente de Grado en Ingeniería Informática, con el objetivo de garantizar la consecución con éxito del presente proyecto.

A continuación, se detallan las EDTs definidas:

- Gestión de proyecto
- Planificación
- Análisis.
- Diseño.

- Implementación.
- Pruebas.
- Plan de implantación.

La siguiente ilustración contiene el diagrama de Gantt del proyecto, en el cual puede observarse el detalle de los EDTs distribuidos en el tiempo, así como los hitos definidos en el apartado anterior.



Nombre	Fecha de inicio	Fecha de fin
Gestión de proyecto	6/03/17	31/05/17
Identificación del problema a resolver	6/03/17	7/03/17
Estado del Arte	8/03/17	8/03/17
Definición Objetivos y Alcance	9/03/17	15/03/17
Seguimiento de proyecto 1	12/03/17	15/03/17
Seguimiento de proyecto 2	10/04/17	12/04/17
Seguimiento de proyecto 3	29/05/17	31/05/17
Cierre y conclusiones	29/05/17	29/05/17
Planificación	8/03/17	15/03/17
Análisis	16/03/17	2/04/17
Descripción del sistema	16/03/17	17/03/17
Requisitos no funcionales	18/03/17	20/03/17
Requisitos funcionales	21/03/17	26/03/17
Definición Roles y Usuarios	27/03/17	29/03/17
Identificación tecnología	30/03/17	2/04/17
Investigación tecnológica	30/03/17	2/04/17
Identificación	3/04/17	3/04/17
Diseño	4/04/17	23/04/17
Casos de Uso	4/04/17	9/04/17
Diseño del Portal	10/04/17	18/04/17
Diseño BBDD	19/04/17	23/04/17
Implementación	24/04/17	10/05/17
Preparación del entorno	24/04/17	26/04/17
Implementación del Portal	25/04/17	6/05/17
Implementación de BBDD	7/05/17	8/05/17
Integración del sistema	8/05/17	10/05/17
Pruebas	11/05/17	20/05/17
Diseño juego de pruebas	11/05/17	13/05/17
Ejecución de pruebas	14/05/17	17/05/17
Corrección de errores	18/05/17	20/05/17
Obtención del producto	21/05/17	21/05/17
Plan de implantación	22/05/17	30/05/17
Análisis de viabilidad	22/05/17	22/05/17
Definición estrategia de implantación	23/05/17	30/05/17

Ilustración 3 Planificación.

1.6.4 Estimación de costes

El presente proyecto se encuentra en el marco del ámbito académico, por lo que los recursos destinados al mismo no pueden cuantificarse a nivel monetario. Sin embargo, durante la fase de planificación si se ha elaborado la estimación de los recursos propios del estudiante a nivel esfuerzo y dedicación horaria, en la siguiente tabla se desglosa el esfuerzo y dedicación estimado para cada uno de los EDTs planteados en la fase de planificación del proyecto.

En la siguiente tabla, es posible observar los recursos estimados para cada uno de los EDTs definidos:

EDT	RECURSOS (horas)
Gestión del proyecto	24
Planificación	24
Análisis	36
Diseño	48
Implementación	60
Pruebas	24
Plan de implantación	24
	240

Ilustración 4 Recursos destinados para cada uno de los EDTs planificados.

1.6.5 Análisis de riesgos

Con el objetivo de garantizar la consecución con éxito del proyecto y prever los principales riesgos potenciales que podrían impactar de manera significativa en el mismo, se han definido un conjunto de riesgos la probabilidad de que estos sucedan, así como el plan de respuesta con tal de mitigar al máximo el impacto que pudieran ocasionar. En la siguiente tabla es posible observar los riesgos definidos:

Riesgo	Impacto	Probabilidad	Medidas Mitigadoras
Estimación horaria poco ajustada en las tareas	Alto	Alta	Aumentar la dedicación horaria a las actividades
Poca experiencia con la tecnología	Medio	Alta	Aumentar la dedicación durante el análisis a la investigación y comprensión de la tecnología escogida
Accidentes/Enfermedades del recurso	Alta	Baja	Replanificar las actividades dentro de la fecha límite establecida y aumentar la dedicación horaria del recurso en los días posteriores
Pérdida de Información del Proyecto	Alto	Baja	Establecer copias de seguridad en sistemas externos que permitan recuperar la información perdida

Ilustración 5 Análisis de riesgos y medidas mitigadoras.

1.7 Producto Obtenido

Una vez introducido el proyecto, en este apartado del Capítulo 1 se procede a describir el producto final obtenido mediante la ejecución de las tareas planificadas y descritas en apartados anteriores.

En este sentido, el prototipo elaborado no es un prototipo que permita su implantación como un sistema final (véase el Anexo 2 Plan de Implantación), sino que consiste en un prototipo mínimo que aporta las funcionalidades definidas y el cual es necesario refinar.

Por lo tanto, la descripción que se aporta a continuación del prototipo obtenido tiene como objetivo aportar al lector una visión de la solución a alto nivel. Para obtener una mayor comprensión de los resultados obtenidos.

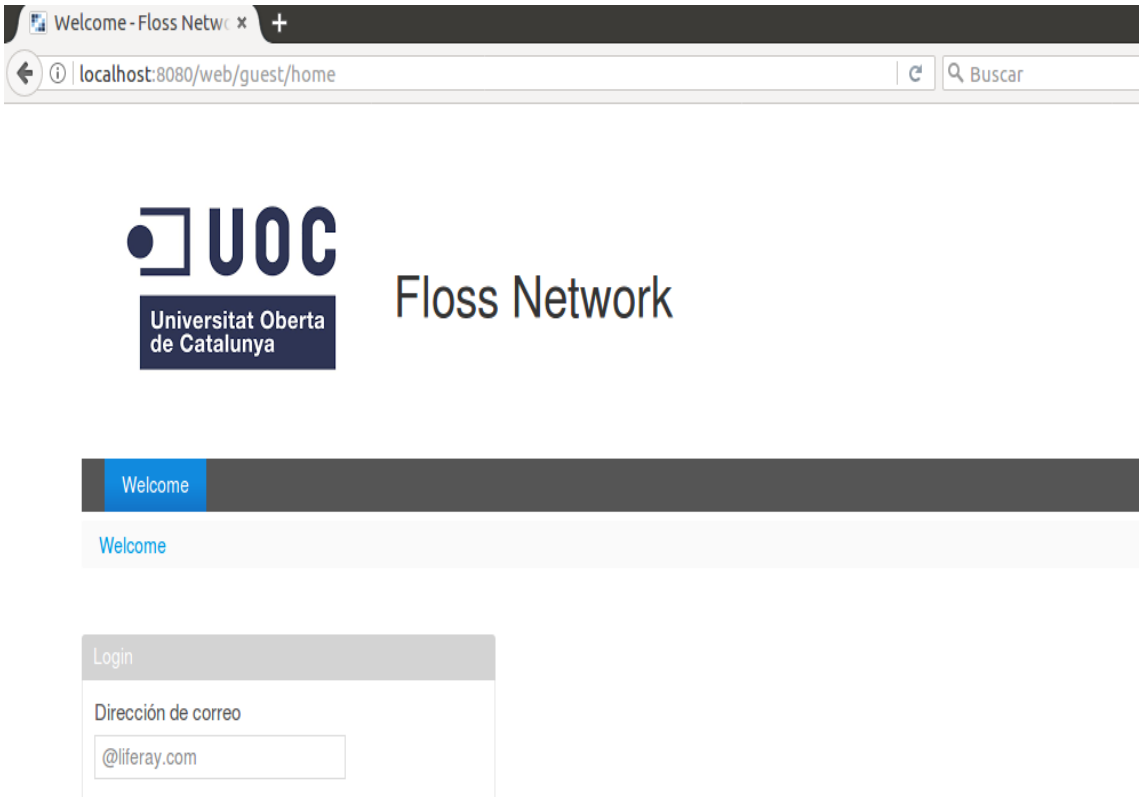


Ilustración 6 Prototipo. Pantalla de inicio.

Se trata de una plataforma del tipo web, a la que se accede al iniciar un servicio Tomcat que despliega los archivos ".war" desarrollados (para conocer más detalle de los desarrollos realizados, véase el Capítulo 3 Implementación de la Plataforma). Los usuarios pueden registrarse o loguearse en la plataforma y acceder a las funcionalidades de esta.

Dependiendo del tipo de usuario registrado (usuario//administrador), la plataforma muestra un menú con las funcionalidades habilitadas, así como las funcionalidades propias configuradas del perfil (la tecnología seleccionada, permite que cada usuario elija los componentes funcionales que desee a los que se le haya habilitado el acceso).

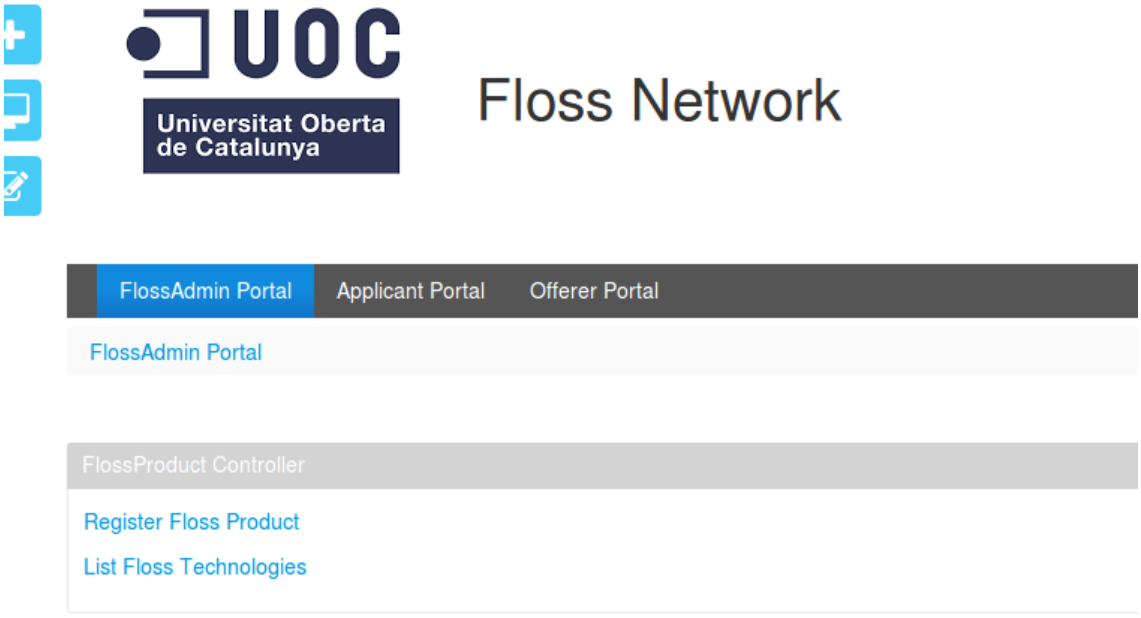


Ilustración 7 Ejemplo de acceso como administrador. Permite cargar un componente para administrar la base de datos de tecnologías del portal, Floss Product.

Si se accede como usuario, permite listar a otros usuarios que se hayan dado como ofertante/demandante, así como permite darse de alta en alguno de estos perfiles para poder disponer de las opciones propias de cada perfil.

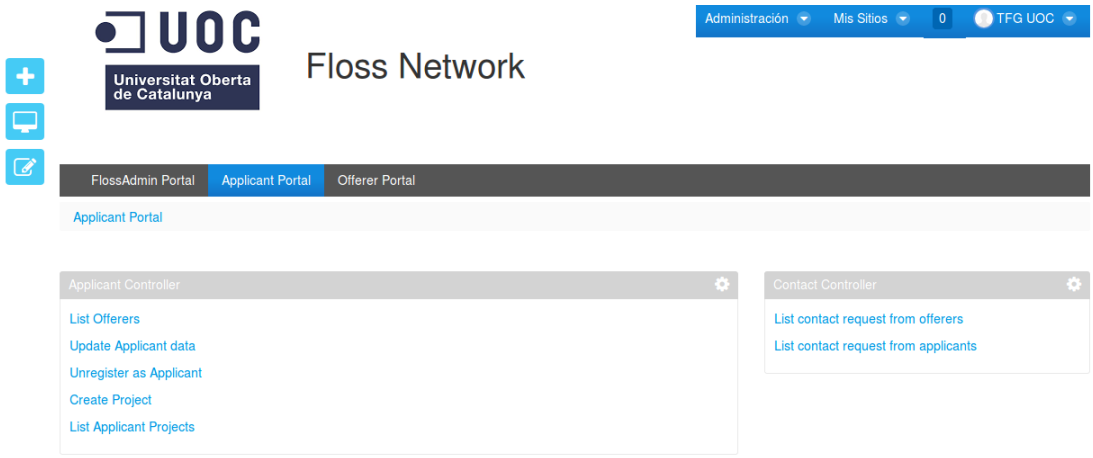


Ilustración 8 Ejemplo funcionalidades disponibles para un usuario dado de alta como demandante.

La interfaz de la solución dispone de un diseño fácil e intuitivo, mediante tecnología *responsive* y *user friendly*. A nivel funcional, como objetivo final del portal, la plataforma ofrece un servicio de que permite a los usuarios registrados y dados de alta en la plataforma con necesidades o experiencia en tecnologías Floss, contactar con otros usuarios que ofrezcan o precisen colaboraciones en un proyecto.

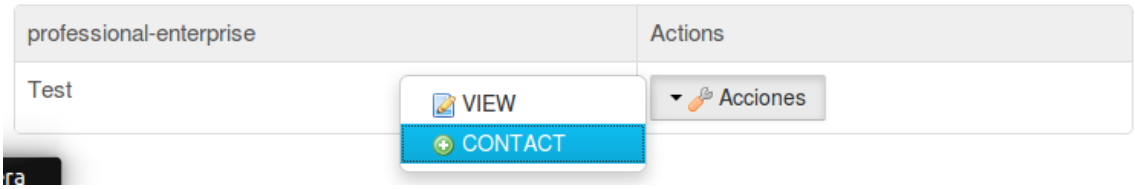


Ilustración 9 Ejemplo de solicitud de contacto a un usuario profesional o empresa que ofrece servicios sobre una tecnología.

A partir de este contacto, el usuario podrá visualizar los datos de contacto del otro, como el correo electrónico, teléfono, etc., así como valorar la colaboración que han establecido con el otro usuario para el proyecto.

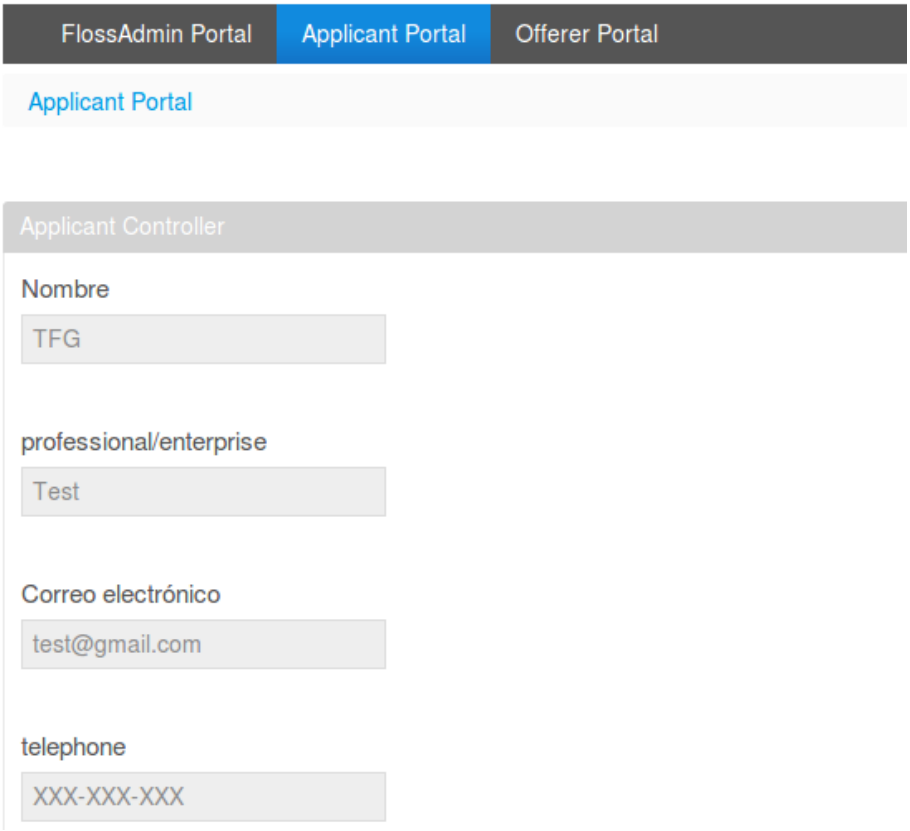


Ilustración 10 El usuario puede visualizar los datos de otros usuarios con los que haya contactado, para dar un primer paso para establecer la colaboración.

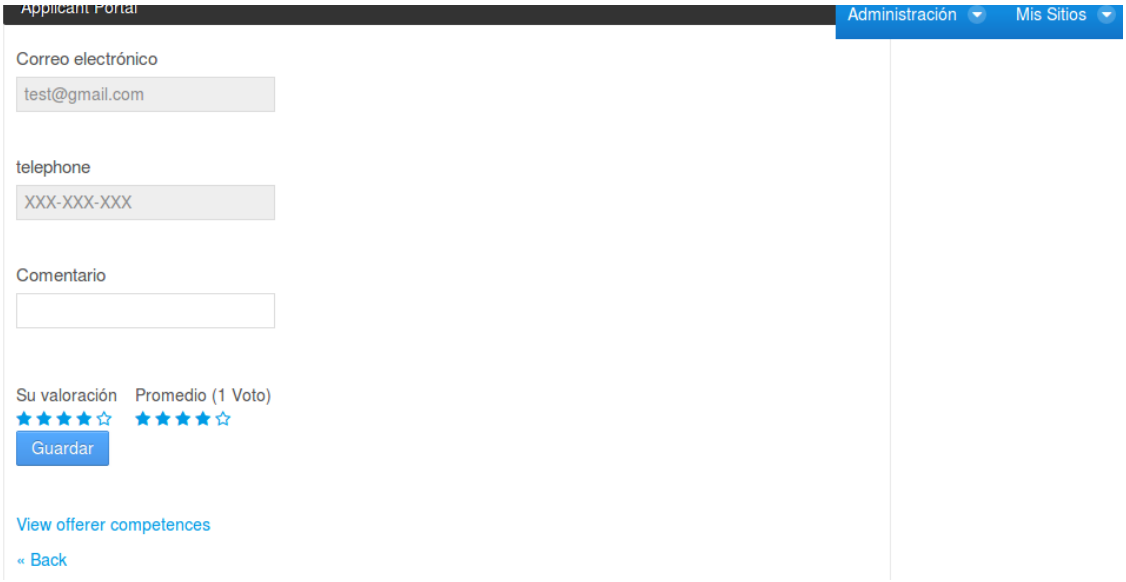


Ilustración 11 La solución permite evaluar una colaboración para un proyecto, mediante puntuación y comentarios para la misma.

1.8 Descripción de otros Capítulos

En este apartado se pretende ofrecer una introducción al resto de capítulos que conforman esta memoria de Trabajo de Fin de Grado. A este respecto, los capítulos describen en un soporte textual las acciones y aspectos más relevantes que se han tomado, así como una descripción más detallada del producto final y las conclusiones extraídas del proyecto.

A continuación, se listan y describen brevemente cada uno de estos capítulos:

- **Capítulo 2**. **Análisis y Diseño**: Este capítulo contiene la descripción y aspectos más relevantes del análisis y diseño de la plataforma. Se describen las actividades realizadas y decisiones que se han tomado en la ejecución de esta fase.
- **Capítulo 3. Implementación de la Plataforma**: Este capítulo contiene la descripción y aspectos más relevantes del desarrollo de la plataforma. Se describen las actividades realizadas y decisiones que se han tomado en la ejecución de esta fase.

- **Capítulo 4. Resultados obtenidos**: En este capítulo se describe y recogen los principales aspectos del prototipo final del proyecto, aportando el detalle del alcance del mismo.
- **Capítulo 5. Conclusiones**: Este capítulo recoge las principales conclusiones extraídas del trabajo realizado, objetivos alcanzados, seguimiento y planificación ejecutados y posibles proyectos derivados.

Capítulo 2. Análisis y Diseño

2.1. Introducción

El objetivo de esta fase ha consistido en la identificación de las necesidades y variables limitantes que debe satisfacer la solución final del proyecto, mediante la descripción del sistema, obtención de requisitos funcionales y no funcionales, definición de roles y usuarios, identificación tecnológica de la solución.

2.2 Descripción del sistema

El portal planteado en el proyecto persigue el desarrollo de una plataforma web que se estructure como punto de referencia, habilitando el contacto entre ofertantes y demandantes de implantaciones de las plataformas, sistemas y soluciones existentes para el mundo de la empresa bajo el paradigma del software libre.

Ilustración 12 Objetivo del portal.

Para seguir la línea de acción del propio portal, se estableció en esta misma fase que la tecnología empleada en el desarrollo de la plataforma fuera del tipo Open Source, alineado con el paradigma de la solución planteada.

Por otro lado, en esta fase se ha definido que la solución resultante deberá permitir principalmente:

- Establecer un canal de comunicación entre ofertantes y demandantes de implantaciones de plataformas, sistemas y soluciones existentes bajo el paradigma del software libre.
- Ofrecer un sistema de valoración y ranking tanto de proveedores como solicitantes, que genere una base de conocimiento de valor añadido para ambos colectivos del portal.
- Consultas de información y flujo de comunicaciones tanto a clientes como proveedores, que permita establecer el primer contacto para el inicio de una colaboración.
- Gamificación de la solución, permitiendo optimizar la experiencia de los usuarios y obtener distinciones que reflejen la reputación digital del cliente/proveedor dentro de la comunidad.

De manera adicional, se ha definido que la plataforma deberá implementar un sistema de registro y login obligatorio para poder acceder a las funcionalidades e información que ofrecerá la solución. Se han definido dos tipos de usuario que podrán darse de alta en el registro:

- **Anónimo**. Cualquier usuario anónimo podrá acceder a la plataforma con la finalidad de poder realizar el login, si se está registrado, y en caso contrario poder acceder al formulario de registro.
- **Demandante**. Los usuarios que accedan a la plataforma podrán loguearse, previo registro en la misma, con el objetivo de crear un proyecto bajo el paradigma FLOSS, consultar información sobre proveedores potenciales para la implementación del proyecto, iniciar el primer contacto para una posible relación profesional con estos, comentar y valorar colaboraciones que se hayan dado con proveedores y publicar ofertas de proyectos de manera que los proveedores interesados puedan contactar con ellos.
- **Ofertante**. Los usuarios que accedan a la plataforma podrán loguearse, previo registro en la misma, con el objetivo de consultar información sobre demandantes potenciales para la implementación de un proyecto software libre de unas determinadas características, iniciar el primer contacto para una posible relación laboral con estos,

comentar y valorar colaboraciones que se hayan dado con demandantes anteriormente y añadir nuevas competencias o aptitudes a su perfil para actualizarlo con novedades.

- **Administrador**: Se ha definido un tipo de usuario adicional, no seleccionable por los usuarios que accedan a la plataforma, que dispondrá de funcionalidades adicionales de administración y mantenimiento de la información y usuarios de la solución. Se plantea que la tecnología escogida ofrezca la posibilidad de realizar funcionalidades o un rol de administrador para la plataforma.

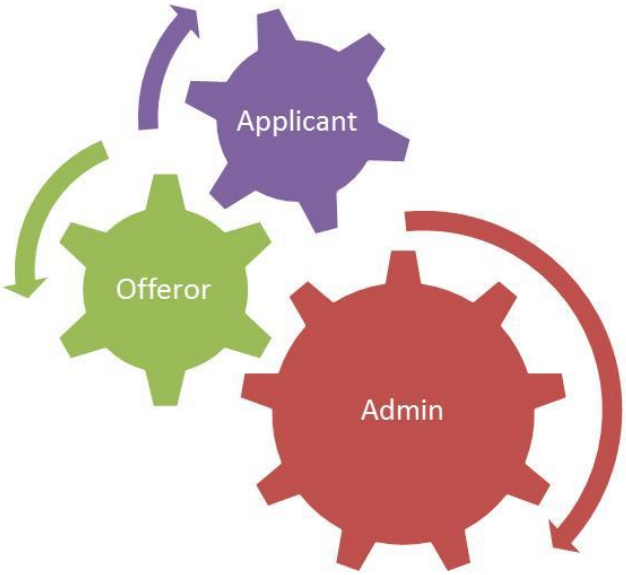


Ilustración 13 Usuarios Definidos.

2.3 Identificación de requisitos

La plataforma planteada gira en torno a un conjunto de funcionalidades que debe ofrecer la solución de proyecto, un prototipo inicial, de manera que sea posible comprobar la operatividad de la misma a un alto nivel, así como valorar posibles proyectos derivados.

Ilustración 14 Operaciones definidas en la plataforma.

En relación a los **ofertantes**, el sistema deberá permitir a esta tipología de usuarios registrar sus competencias dentro del catálogo de productos FLOSS definido, ofrecer a los demandantes sus habilidades y competencias, enviar solicitudes de contacto a demandantes que se encuentren registrados en la plataforma, así como valorar las colaboraciones realizadas.

En referencia a los **demandantes**, la plataforma permitirá a estos usuarios definir un proyecto bajo el paradigma FLOSS, para el cual detallarán las necesidades, características y alcance del mismo (p.ej.: "Proyecto con tecnología Odoo para el cual es necesario desarrollar a medida funcionalidades de ventas y gestión de Inventario. Presupuesto máximo 60.000€), contactar con ofertantes que estén registrados en la plataforma, así como valorar las colaboraciones realizadas.

Durante la ejecución de esta fase, inicialmente se planteó la obligatoriedad de la coincidencia entre la competencia sobre una tecnología FLOSS de un ofertante y el proyecto registrado por un demandante para iniciar el contacto entre ambos. No obstante, se descartó esta restricción en la operativa para permitir el contacto, no únicamente entre proyectos que estén a punto de iniciarse, sino que se habilite el

mismo para futuros proyectos que no estén definidos sobre cualquier tecnología FLOSS.

Las distintas operaciones y la funcionalidad de la plataforma se han definido por el conjunto de requisitos funcionales y no funcionales del proyecto. A continuación, se detalla cada una de las necesidades identificadas y derivadas del sistema definido, así como sus características, mediante la especificación formal de requisitos funcionales y no funcionales del proyecto.

Requisitos funcionales

Código	**RF1**
Nombre	**Acceso a la aplicación**
Criticidad	**10**
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional
Descripción del requisito	Cualquier usuario que acceda al portal, tendrá acceso a la información general del portal, así como a loguearse, registrarse y realizar la búsqueda de ofertantes y demandantes y visualizar el listado de usuarios y de productos floss que contempla el portal para contacto entre demandantes y ofertantes
Entrada	
Salida	Pantalla inicial de la aplicación
Restricciones	Lo usuarios no necesitarán darse de alta como ofertante o demandante (registro sí es obligatorio) para listar a otros usuarios, pero no debe permitirse ver los datos de contacto.

Código	**RF2**
Nombre	**Registro/Login en la aplicación**
Criticidad	**8**
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional
Descripción del requisito	Cualquier usuario que acceda al sistema podrá realizar el registro en el mismo, a partir del cual aportarán la información necesaria para darse de alta en la plataforma y loguearse en la misma.
Entrada	Nombre de Usuario; Password
Salida	Usuario con acceso al sistema
Restricciones	El usuario no debe constar como dado de alta en el sistema

Código	**RF3**
Nombre	**Alta Ofertante**
Criticidad	**8**
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional
Descripción del requisito	El sistema deberá implementar una función de alta que permita a los usuarios registrarse como ofertante en la plataforma.
Entrada	Nombre profesional/empresa; email; teléfono; población.
Salida	Usuario dado de alta como ofertante en la plataforma
Restricciones	El usuario debe estar registrado en la plataforma El usuario debe estar logueado en la plataforma El usuario no debe constar como ofertante en la plataforma

Código	**RF4**
Nombre	**Alta Demandante**
Criticidad	**8**
Fuente del requisito	Formulario de ingreso de datos
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional
Descripción del requisito	El sistema deberá implementar una función de alta que permita a los usuarios registrarse como demandante en la plataforma
Entrada	Nombre profesional/empresa; email; teléfono; población.
Salida	Usuario dado de alta como demandante en la plataforma
Restricciones	El usuario debe estar registrado en la plataforma El usuario debe estar logueado en la plataforma El usuario no debe constar como demandante en la plataforma

Código	**RF5**
Nombre	**Registro/Edición de competencias**
Criticidad	**8**
Fuente del requisito	Formulario de ingreso de datos
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional
Descripción del requisito	El sistema debe permitir a los usuarios registrar/editar sus competencias en productos FLOSS.
Entrada	Listado con productos FLOSS dados de alta en la plataforma
Salida	El usuario tiene asociada la competencia a uno o más productos FLOSS
Restricciones	El usuario debe estar registrado previamente en la plataforma. El usuario debe estar logueado en la plataforma. El usuario debe estar dado de alta como ofertante.

	El usuario puede seleccionar como mucho 4 competencias en productos FLOSS.

Código	**RF6**
Nombre	**Registro/Edición de proyecto**
Criticidad	**8**
Fuente del requisito	Formulario de ingreso de datos
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional
Descripción del requisito	El sistema debe permitir a los usuarios demandantes registrar/editar un proyecto asociado a un producto FLOSS del catálogo de productos. El usuario deberá indicar si el proyecto se encuentra activo o no.
Entrada	Formulario de ingreso de datos con productos FLOSS dados de alta en la plataforma
Salida	El usuario ha registrado un nuevo proyecto asociado a su propio usuario y a un producto FLOSS del listado
Restricciones	El usuario debe estar registrado previamente en la plataforma. El usuario debe estar logueado en la plataforma. El usuario debe estar dado de alta como demandante. El usuario no podrá eliminar un proyecto que disponga de un comentario/valoración de otro usuario.

Código	**RF7**
Nombre	**Actualizar/Modificar datos**
Criticidad	**8**
Fuente del requisito	Formulario de ingreso de datos
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional
Descripción del requisito	El sistema debe permitir a los usuarios actualizar sus datos de contacto registrados en la plataforma
Entrada	Nombre profesional/empresa; email; teléfono; población.
Salida	Pantalla de confirmación de actualización de datos.
Restricciones	El usuario debe estar registrado en la plataforma El usuario debe estar dado de alta como demandante/ofertante El usuario debe estar logueado en la plataforma

Código	**RF8**
Nombre	**Registro/Edición de productos FLOSS**

Criticidad	**8**
Fuente del requisito	Formulario de ingreso de datos
Prioridad del requisito	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/ Opcional
Descripción del requisito	El sistema deberá permitir añadir, editar y eliminar el listado de Productos FLOSS de la plataforma
Entrada	Listado de productos FLOSS
Salida	Listado actualizado de productos FLOSS
Restricciones	La modificación de Competencias (añadir/eliminar) deberá ser ejecutada por el usuario administrador de la plataforma

Código	**RF9-1**
Nombre	**Consultar ofertantes**
Criticidad	**9**
Fuente del requisito	Formulario de ingreso de datos
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional
Descripción del requisito	El sistema debe permitir a los usuarios ver un listado de ofertantes registrados en la plataforma
Entrada	El sistema deberá permitir a los usuarios visualizar el listado de ofertantes registrados en la plataforma.
Salida	Pantalla con el listado de ofertantes
Restricciones	Los datos de contacto no deben mostrarse.

Código	**RF9-2**
Nombre	**Consultar demandantes**
Criticidad	**9**
Fuente del requisito	Formulario de ingreso de datos
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional
Descripción del requisito	El sistema debe permitir a los usuarios ver un listado de demandantes registrados en la plataforma
Entrada	El sistema deberá permitir a los usuarios visualizar el listado de demandantes registrados en la plataforma
Salida	Pantalla con el listado de demandantes
Restricciones	Los datos de contacto no deben mostrarse

Código	**RF10**
Nombre	**Iniciar contacto**
Criticidad	**9**
Fuente del requisito	Formulario de ingreso de datos
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

Descripción del requisito	El sistema, debe permitir al usuario demandante enviar una solicitud de contacto a un ofertante, así como permitir a un ofertante enviar una solicitud de contacto al demandante. Esta solicitud otorgará acceso ambos a los datos de contacto del otro.
Entrada	Formulario de ingreso de datos
Salida	Confirmación de envío de solicitud
Restricciones	El usuario debe estar logueado en la aplicación. El usuario debe haber listado un conjunto de demandantes/ofertantes.

Código	**RF11**
Nombre	**Aceptar contacto**
Criticidad	**9**
Fuente del requisito	Formulario de ingreso de datos
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional
Descripción del requisito	El sistema debe permitir a los usuarios aceptar /rechazar solicitudes de contacto enviadas por otros usuarios. Si la solicitud es aceptada, el solicitante y el usuario podrán visualizar los datos de contacto del otro y se generará un contacto vinculado a ambos usuarios.
Entrada	Solicitud de contacto
Salida	Confirmación de contacto
Restricciones	El usuario debe estar logueado en la aplicación. El usuario debe haber recibido una solicitud de contacto

Código	**RF12**
Nombre	**Valorar Contacto**
Criticidad	**8**
Fuente del requisito	Formulario de ingreso de datos
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional
Descripción del requisito	Cuando la plataforma ha habilitado un contacto para un proyecto, se deberá introducir la valoración del proyecto (ofertante) y la valoración de adecuación de la competencia (demandante), y está constará como pendiente hasta realizarla. Se podrán introducir diversas valoraciones para un mismo contacto.
Entrada	Puntuación; Comentario
Salida	Pantalla de confirmación de valoración
Restricciones	El usuario debe estar logueado en la aplicación. El usuario debe haber accedido a un ofertante/demandante El ofertante/demandante debe haber aceptado la solicitud de contacto El usuario podrá realizar distintas valoraciones para un contacto, pero no eliminar las mismas.

Código	**RF13**
Nombre	**Contactar Usuario**
Criticidad	**10**
Fuente del requisito	Formulario de ingreso de datos
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional
Descripción del requisito	El sistema debe permitir a los usuarios acceder a los datos de contacto de otros usuarios, una vez hayan establecido el contacto inicial (aceptación de iniciar contacto).
Entrada	
Salida	Pantalla con datos de contacto
Restricciones	El usuario debe estar logueado en la aplicación. El usuario debe haber establecido el contacto para un proyecto con un ofertante/demandante.

Código	**RF14**
Nombre	**Obtención de insignia de reputación**
Criticidad	**3**
Fuente del requisito	Valoración de usuario
Prioridad del requisito	☐ Alta/Esencial ☐ Media/Deseado ☒ Baja/ Opcional
Descripción del requisito	El sistema otorgará una insignia de reputación al usuario demandante/ofertante para cada proyecto asociado a un contacto, con la media de valoraciones obtenidas para este
Entrada	Valoraciones de usuarios
Salida	Asignación de insignia
Restricciones	El usuario debe encontrarse registrado en la plataforma El usuario debe estar dado de alta como ofertante/demandante El usuario debe haber iniciado el contacto para un proyecto. El usuario debe haber recibido al menos una valoración para el contacto

Código	**RF15**
Nombre	**Visualización de insignia**
Criticidad	**3**
Fuente del requisito	Valoración de usuario
Prioridad del requisito	☐ Alta/Esencial ☐ Media/Deseado ☒ Baja/ Opcional
Descripción del requisito	El sistema debe permitir visualizar las insignias asignadas a un usuario.
Entrada	Selección de usuario
Salida	Pantalla con insignias asignadas

Restricciones	El usuario debe estar logueado. El sistema solo debe mostrar insignias de ofertantes a usuarios demandantes El sistema solo debe mostrar insignias de demandantes a usuarios ofertantes

Requisitos no funcionales

Código	**RNF1**
Nombre	**Curva de Aprendizaje**
Criticidad	**6**
Fuente del requisito	Usuarios
Prioridad del requisito	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/ Opcional
Descripción del requisito	Un nuevo usuario debe ser capaz de usar el sistema sin dedicar mucho esfuerzo a aprender su funcionamiento.

Código	**RNF2**
Nombre	**Escalabilidad**
Criticidad	**5**
Fuente del requisito	Implementación
Prioridad del requisito	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/ Opcional
Descripción del requisito	El sistema resultante debe ser escalable y permitir definir y añadir nuevas funcionalidades en un futuro

Código	**RNF3**
Nombre	**Facilidad de integración BBDD**
Criticidad	**10**
Fuente del requisito	Desarrollo/BBDD
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional
Descripción del requisito	El sistema debe permitir la integración con diferentes gestores de BBDD y su configuración debe ser fácil e intuitiva

Código	**RNF4**
Nombre	**OpenSource**
Criticidad	**10**
Fuente del requisito	Tecnología
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional
Descripción del requisito	La tecnología sobre la cual debe desarrollarse la solución debe ser del tipo software libre

Código	**RNF5**
Nombre	**Componentes**
Criticidad	**10**
Fuente del requisito	Tecnología
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional
Descripción del requisito	La tecnología sobre la cual debe desarrollarse la solución debe permitir el desarrollo de componentes en lenguajes orientados a objetos (conocidos por el autor del proyecto) que permitan implementar funcionalidades o futuras funcionalidades que no disponga la plataforma por si misma

Código	**RNF6**
Nombre	**Interfaces Gráficas**
Criticidad	**6**
Fuente del requisito	Diseño/Desarrollo
Prioridad del requisito	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/ Opcional
Descripción del requisito	La tecnología sobre la cual debe desarrollarse la solución debe permitir el uso de plantillas para el diseño y creación de la interfaz gráfica de la solución. Las interfaces deben estar implementadas con tecnología responsiva para permitir su adaptación, independientemente del dispositivo desde el que se acceda

Código	**RNF7**
Nombre	**Tiempo de respuesta**
Criticidad	**7**

Fuente del requisito	Calidad
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional
Descripción del requisito	La solución debe mostrar tiempos de respuesta rápido ante las distintas peticiones de los usuarios

Código	**RNF8**
Nombre	**Navegadores**
Criticidad	**9**
Fuente del requisito	Desarrollo
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional
Descripción del requisito	El sistema de información debe ser web, y permitir su ejecución con diferentes navegadores web.

Código	**RNF9**
Nombre	**Madurez Tecnológica**
Criticidad	**9**
Fuente del requisito	Desarrollo
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional
Descripción del requisito	La plataforma debe mostrar una madurez tecnológica elevada

2.4 Análisis de ítems

A partir de los requerimientos formalizados en el apartado anterior, es posible determinar principales elementos que compondrán el sistema y las relaciones que existen entre estos.

En este apartado, se procede a analizar y detallar los ítems en cuestión.

2.4.1 Usuarios del sistema

De la definición del sistema y los requisitos establecidos, es posible identificar tipos de usuario, los cuales son analizados a continuación:

Usuario Anónimo

Se ha definido y formalizado un tipo de usuario para todos los usuarios que accedan al sistema. En este sentido, una vez un usuario accede al sistema, tendrá permitido, o bien registrarse y loguearse en la plataforma.

Usuario Demandante

Se ha definido y formalizado un tipo de usuario que representará a los usuarios Demandantes una vez estos se hayan logueado en el sistema. Una vez se haya logueado en el sistema, podrá:

- Acceder a los usuarios ofertantes para visualizar las competencias de estos.
- Enviar una solicitud de contacto en el caso que pudiera interesar.
- Del mismo modo, podrá aceptar peticiones de contacto enviadas por otros usuarios.
- Listar los usuarios ofertantes/demandantes y visualizar datos genéricos de estos.
- Visualizar los datos de contacto de contactos iniciados.
- Registrar nuevos proyectos asociados a productos del catálogo FLOSS.
- Realizar valoración de contactos establecidos previamente.
- Actualizar la información de perfil (incluyendo proyecto demandado).
- Visualizar Insignias de contactos.

Usuario Ofertante

Se ha definido y formalizado un tipo de usuario que representará a los usuarios Ofertantes una vez estos se hayan logueado en el sistema. Una vez realizado el login, podrán:

- Acceder a los usuarios demandantes para visualizar los proyectos definidos por estos.
- Enviar una solicitud de contacto en el caso que pudiera interesar.
- Del mismo modo, podrá aceptar peticiones de contacto enviadas por otros usuarios.
- Listar los usuarios ofertantes/demandantes y visualizar datos genéricos de estos.
- Visualizar los datos de contacto de contactos iniciados.
- Realizar valoración de contactos establecidos previamente.
- Actualizar la información de perfil (incluyendo las competencias del ofertante, dentro del catálogo de FLOSS definido en la plataforma).
- Visualizar Insignias de contactos.

FLOSS Product

La plataforma que plantea la solución del proyecto incluye un catálogo de tecnologías de código libre y abierto (FLOSS), que estará definida por el administrador de la misma.

En este sentido, el administrador deberá especificar los datos necesarios de cada producto o tecnología FLOSS, de manera que la identificación de la misma sea inequívoca por los usuarios de la plataforma. De este modo, los atributos mínimos a especificar serán:

- Nombre.
- Versión.
- Tipo de aplicación (Cloud/Cliente-Servidor).
- Lenguaje de programación.
- BBDD compatibles.

Contacto

Dado un usuario demandante que haya registrado un proyecto asociado a un producto FLOSS, el sistema deberá permitir que usuarios ofertantes puedan enviar una solicitud de contacto al demandante, o al demandante enviar una solicitud de contacto a ofertantes con competencias en el proyecto.

Una vez la solicitud de contacto haya sido aceptada, tanto el ofertante como el demandante deberán valorar la colaboración (Contacto).

Valoración

Los usuarios deberán poder realizar la valoración de los distintos contactos realizados con otros usuarios. Esta valoración quedará en “Estatus” pendiente, hasta que se haya realizado. En este sentido, la valoración permitirá definir el grado de satisfacción de los usuarios con las colaboraciones que hayan efectuado con otros usuarios.

La valoración quedará definida por la ponderación del resultado de una encuesta de satisfacción generada a partir del contacto entre dos usuarios. De este modo, los usuarios deberán valorar del 1 al 5, las siguientes cuestiones (según sea demandante u ofertante):

- Valore el grado de satisfacción con la solución obtenida/adecuación de la descripción a la solución final.
- Valore el grado de satisfacción con la gestión del proyecto por parte del demandante/ofertante.
- ¿Recomendaría a otros usuarios la colaboración con el ofertante/demandante?

A partir de las valoraciones del usuario, se ponderaría la ratio de valoración otorgándole una valoración final al ofertante/demandante.

Ficha proyecto/competencia

Los usuarios demandantes deberán definir el proyecto que plantea la colaboración en un campo de texto libre. En este campo, deberían constar la tecnología del proyecto, las necesidades definidas a un alto nivel (módulos/funcionalidades necesarias), la fecha estimada para el inicio de la colaboración y el presupuesto máximo del proyecto.

Por otro lado, los usuarios ofertantes podrán aportar de manera adicional un campo en el que incluya la experiencia con el producto FLOSS en cuestión, así como sus principales logros y aptitudes para abordar proyectos con la tecnología.

Plataforma

El sistema de información identificado será la plataforma base que sustentará el sistema. A partir de los requisitos no funcionales identificados. En este sentido, para el análisis de la plataforma, se procede a detallar algunas alternativas del tipo OpenSource (el requisito NF que define este tipo de plataformas acota el análisis) que se han tenido en cuenta para el análisis e identificación de la plataforma base.

- **Joomla**. Solución de gestión de contenidos OpenSource que permite la implementación de sitios web dinámicos e interactivos. Está desarrollado en PHP y requiere para su uso un gestor de BBDD, así como un servidor HTTP de apache.
- **Drupal**. Solución de gestión de contenidos OpenSource, modular, multipropósito y configurable. Permite el uso de servicios añadidos como encuestas, votaciones y administración de usuarios y permisos. Está desarrollado en PHP y necesita un gestor de BBDD del tipo MySQL.
- **Liferay**. Solución de gestión de contenidos OpenSource, desarrollado en Java. Permite correr en la mayoría de servidores de aplicaciones y contenedores servlets, base de datos y sistemas operativos. Dispone de servicios de administración de usuarios,

atributos personalizados, correo electrónico, layouts responsivas, diseño por temas, creación de directorios.

En la siguiente matriz, puede verse una comparativa de las distintas soluciones identificadas respecto al cumplimiento de los requisitos definidos:

Requisito/Plataforma	Drupal	Joomla	Liferay Portal	
RNF1	0,6	0,8	0,8	
RNF2	1	1	0,8	
RNF3	0,8	0,8	1	
RNF4	1	1	1	
RNF5	0	0	1	
RNF6	0,6	0,4	0,6	
RNF7	1	1	1	
RNF8	0,8	1	1	
RNF9	0,8	1	1	
RF1	1	1	1	
RF2	0,6	1	1	
RF3	1	1	1	
RF4	1	1	1	
RF5	0	0	0,5	*El desarrollo de componentes debe ser en java o pyhton, conocidos por el autor
RF6	0	0	0,8	*El desarrollo de componentes debe ser en java o pyhton, conocidos por el autor
RF7	1	1	1	
RF8	0	0	0,5	
RF9	1	1	1	
RF10	0	0	1	
RF11	0	0	1	
Nota FINAL	**12,2**	**13**	**18**	

Ilustración 15 Matriz de cumplimiento de requisitos.

Tal y como se ha comentado, la comparativa se ha llevado a cabo en función de los pesos de criticidad de los requisitos definidos y el nivel en el cual la plataforma permite satisfacer estos.

Las opciones de Drupal y Joomla sobre un lenguaje no orientado a objetos, hacen que estas soluciones queden en desventaja frente a Liferay. Por otro lado, para la integración con el mayor número de bases de datos las soluciones son compatibles con:

- **Drupal**: MySQL, SQL Server y Oracle.
- **Joomla**: MySQL.
- **Liferay**: MySQL, Oracle y PosqreSQL.

Dado que la idea es trabajar sobre una solución del tipo OpenSource, donde una de las BBDD más destacadas es PostgreSQL, este punto ha impactado negativamente en la valoración de Drupal y Joomla.

Finalmente, la integración con los navegadores y la madurez tecnológica de la solución, hacen que **la tecnología finalmente escogida para la plataforma base de la solución haya sido Liferay** (véase el apartado **2.5.2 Marco Tecnológico**, para obtener mayor detalle de la plataforma).

2.5 Diseño del Portal

Durante la etapa de análisis, se han identificado los principales ítems de la plataforma, así como el marco tecnológico sobre el cual se ha trabajado en la fase de desarrollo del proyecto.

No obstante, con el objetivo que el lector pueda disponer de un mayor detalle del diseño funcional de la solución, a continuación, se aportan los diagramas elaborados durante la fase de diseño y los componentes tecnológicos que ofrece la plataforma para la implementación de la lógica de negocio.

2.5.1 Diagramas

Con el objetivo de modelar el análisis de ítems elaborado y funcionalidades definidas, a continuación, se muestra el diagrama de componentes de la solución planteada.

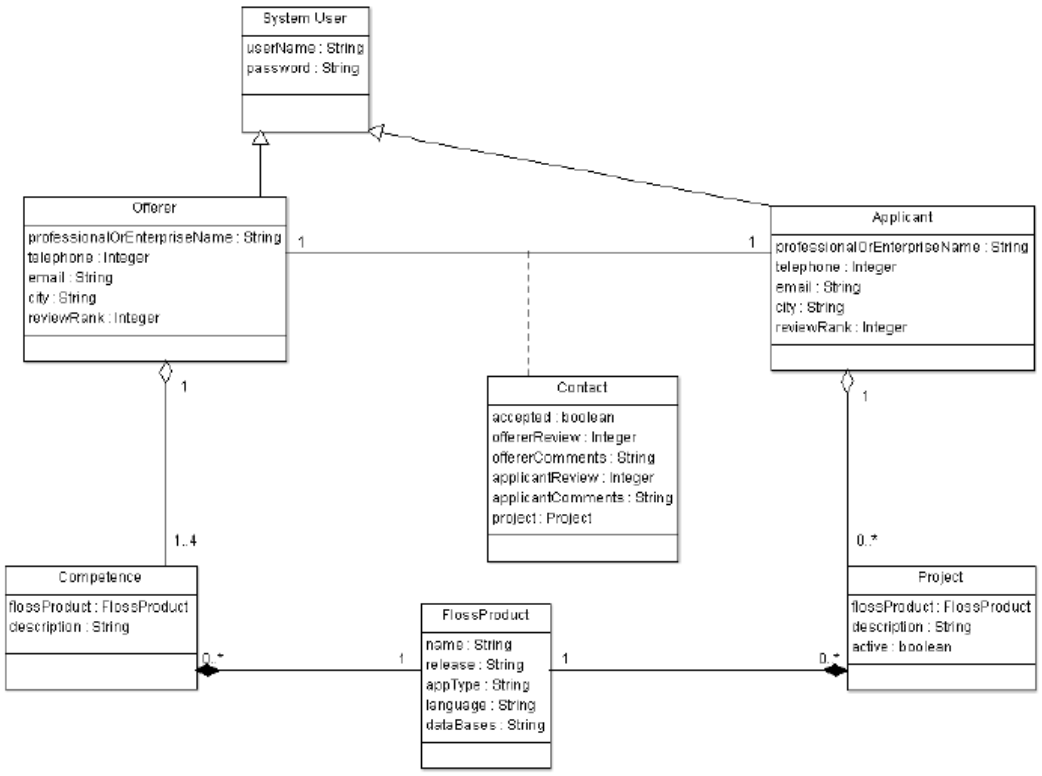


Ilustración 16 Diagrama de entidades de la solución.

En el diagrama es posible observar las distintas entidades que intervendrán en el sistema, la información que es necesaria mantener o almacenar para cada una de estas, así como la interrelación entre las distintas entidades.

De este modo, es posible observar que:

- Se han definido tres tipos de entidades, **System User, Applicant y Offerer**, que representarán los usuarios con distintos roles del sistema (el rol de administrador lo provee la propia plataforma, tal y como se explica en el siguiente apartado). Para estas entidades es necesario almacenar los datos propios de cada usuario, siendo estos:
    - userName: que almacena el nombre de usuario.
    - password: que almacena la contraseña de usuario.
    - professionalOrEnterprise: que almacena el nombre de la empresa del usuario.
    - telephone: que almacena el número de teléfono de contacto del usuario.

    - email: que almacena el correo de contacto del usuario.
    - reviewRank: que almacena la valoración global para el usuario registrada por otros usuarios de la plataforma.
- Se ha definido la entidad **Contact** que nace de la relación entre dos usuarios de la plataforma, un ofertante y un demandante, a raíz de la solicitud de contacto entre estos. Para esta entidad es necesario almacenar los siguientes datos:
    - accepted: que permite conocer si la solicitud de contacto entre un ofertante y un demandante ha sido aceptada.
    - offererReview: que permite almacenar la puntuación otorgada por el usuario ofertante a la colaboración.
    - applicantReview: que permite almacenar la puntuación otorgada por el usuario demandante a la colaboración.
    - applicantComments: que permite almacenar el comentario del demandante sobre la colaboración.
    - offererComments: que permite almacenar el comentario del ofertante sobre la colaboración.
    - project: que permite relacionar el contacto con el proyecto registrado por el usuario demandante.
- Se ha definido la entidad **Compentece**, que permite almacenar y relacionar las competencias de un usuario ofertante sobre distintas tecnologías FLOSS.
- Se ha definido la entidad **Project**, que representa los proyectos registrados por un demandante sobre una tecnología FLOSS.
- Se ha definido la entidad **FlossProduct**, que representa las tecnologías FLOSS registradas en la plataforma y para las cuales los usuarios ofertantes y demandantes pueden registrar o bien competencias, o bien proyectos respectivamente.

De manera adicional, se ha elaborado el siguiente diagrama de casos de uso que permite aportar un mayor detalle de las distintas funcionalidades definidas y los distintos actores que intervienen en el sistema.

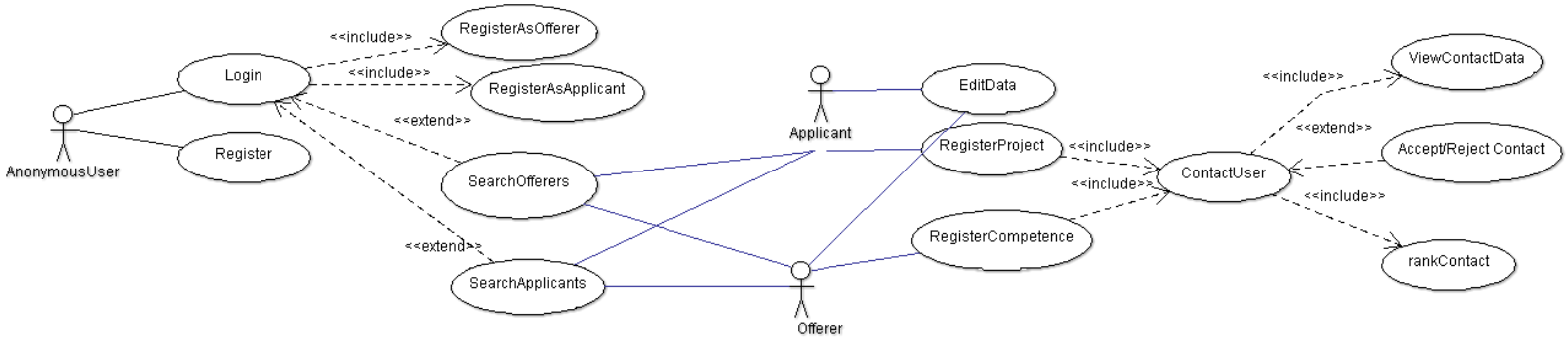


Ilustración 17 Diagrama de casos de Uso.

Para cada uno de los actores del sistema las funcionalidades que puede ejecutar son:

- **AnonymousUser**. Se trata de un usuario que ha accedido al sistema, pero no se ha logueado no el mismo. Las funcionalidades disponibles para este actor son:
    - Login. Permite al usuario acreditarse en la plataforma y acceder a las funcionalidades asociadas a su rol.
    - Register. Permite al usuario registrarse en la plataforma y disponer de datos o credenciales de acceso al mismo.

  Una vez que un usuario anónimo se encuentra registrado, pasa a tener el rol de "System User", pudiendo realizar las siguientes acciones:

    - Registrarse como Ofertante.
    - Registrarse como Demandante.
    - Listar Ofertantes.
    - Listar Demandantes.
- **Applicant**. Se trata de un usuario que ha accedido al sistema y se encuentra registrado como demandante. Las funcionalidades disponibles son:
    - Modificar sus datos en el sistema.
    - Registrar un proyecto sobre una tecnología del catálogo FLOSS.
    - Listar ofertantes y enviar solicitudes de contacto.
    - Aceptar/Rechazar solicitudes de contacto enviadas por otros usuarios.
    - Visualizar datos de contacto de aquellos usuarios con los que se haya establecido el contacto previamente.
    - Valorar y comentar las colaboraciones con otros usuarios que se haya contactado.
- **Offerer**. Se trata de un usuario que ha accedido al sistema y se encuentra registrado como ofertante. Las funcionalidades disponibles son:
    - Modificar sus datos en el sistema.
    - Registrar sus competencias sobre una tecnología del catálogo FLOSS.
    - Listar demandantes y enviar solicitudes de contacto.
    - Aceptar/Rechazar solicitudes de contacto enviadas por otros usuarios.
    - Visualizar datos de contacto de aquellos usuarios con los que se haya establecido el contacto previamente.
    - Valorar y comentar las colaboraciones con otros usuarios que se haya contactado.

2.5.2 Marco Tecnológico

Durante la fase de diseño del portal, se ha analizado la plataforma Liferay, que pretende ser la base de la solución planteada el presente proyecto. A continuación, se procede a detallar las características de la plataforma, principalmente en aquellos aspectos que satisfacen los requerimientos y necesidades definidos, que permitirán sentar las bases de conocimiento al lector para entender el alcance de la solución.

La tecnología *Liferay Portal*, basa su funcionamiento en sistemas del tipo *Enterprise Portal Software* (EPS), del tipo *Open Source*. Tiene como finalidad focalizar los esfuerzos en el contenido y entrega de aplicaciones, ofrecer contenido personalizado o customizado a los usuarios, permitir la gestión y control del acceso a contenidos y aplicaciones, contextualizar grandes conjuntos de contenidos, así como permitir la integración con distintos *frameworks* disponibles bajo el paradigma de *software* libre.

Así mismo, *Liferay Portal* integra la solución *Liferay CMS,* un Sistema de Gestión de Contenido (*CMS, Content Management System*, en inglés). Este tipo de herramientas tienen como finalidad de uso la creación, edición, gestión y publicación contenido digital multimedia en diversos formatos. Para ello, genera páginas web dinámicas interactuando con el servidor web que ofrece la página bajo petición del usuario, con el formato predefinido y el contenido extraído de la base de datos del servidor.

En su conjunto, se trata de una herramienta de código abierto desarrollada en Java, que permite ofrecer un entorno web colaborativo y sistema base para portales de fácil uso, gracias a una interfaz de usuario intuitiva.

Permite facilitar la gestión de la información del servidor gracias a su funcionamiento bajo formatos estandarizados, reduciendo el tamaño de las páginas para descarga y el coste de gestión del portal con respecto a un sitio web estático en el que cada cambio de diseño debe ser realizado en todas las páginas web.

Liferay plantea que, a partir de una o más plantillas diseñadas, desde su panel de control se generen páginas y contenido web dinámico con su interfaz gráfica basada en estas mismas. Además, permite que cada usuario que acceda a una plataforma desarrollada con Liferay pueda gestionar y configurar sus propias páginas, añadiendo, eliminando o configurando las aplicaciones disponibles en esta.

De este modo, como sistema *EPS*, permiten el desarrollo ágil de un portal web principalmente mediante:

- **Herramientas de diseño de interfaces de usuario**. Liferay Portal permite agilizar el desarrollo ágil de un portal simplificando la elaboración de sitios web internos, externos y de canal, unificando el acceso en un único punto con una interfaz de usuario amigable.
- **Framework de integración de aplicaciones**. Liferay Portal ofrece un punto de acceso centralizado que permite a usuarios, administradores y desarrolladores integrar contenido y aplicaciones *backend* y *legacy*, incluyendo distintos métodos de integración (p.ej.: SOAP, REST, RSS, así como APIs propietarias).
- **Soporte de Single Sign On (SSO)**. Permite ofrecer un único punto de acceso a distintos contenidos y aplicaciones, facilitando el acceso desde una única sesión a los usuarios.
- **Campos personalizados**. Ofrece a los administradores la capacidad de personalizar, editar, añadir y modificar campos del perfil de usuarios, así como ofrecer esta funcionalidad para páginas, contenidos web, entradas de blogs, etc.
- **Integración de motores de reglas**. Ofrece la capacidad de personalizar las funcionalidades y contenidos específicos a usuarios, en función de su ubicación, actividad y otros atributos del perfil.
- **Plataforma SOA**. Con el desarrollo de la plataforma siguiendo una arquitectura orientada a servicios, la plataforma permite la integración con distintos sistemas y aplicaciones corporativas de manera ágil y sencilla.
- **Framework de Workflow**. La plataforma incorpora un motor de *workflow* integrado que le permite definir procesos de publicación y aprobación basados en sus necesidades concretas de negocio y operaciones.

Los componentes que conforman las funcionalidades de la plataforma y permiten establecer un portal único de acceso son:

- ***Portlet***. Se trata de aplicaciones contenidas dentro de un portal y se encargan de generar contenidos dinámicos para él. Existen dos especificaciones, JSR 168 y JSR 286, que definen una serie de funciones y clases que deben implementar las aplicaciones para poder seguir el estándar y de esa forma, ser portables de un portal a otro.

A nivel técnico, un *portlet* el desarrollo de un portlet se inicia con una clase *Java* (Liferay dispone de un Software Development Kit, SDK, y un plugin para Eclipse) que implementa la clase interfaz *Java*x.*portlet*.*Portlet*, y contiene la lógica de negocio o funcionalidad de la solución. Permite interactuar con los usuarios web con un paradigma de peticiones y respuestas (*request/response*).

Los *portlet* no pueden ser llamados de manera directa, sino que Liferay obliga a que las llamadas se realicen sobre una página, que pueden contener uno o más Portlets cuyo contenido se genera de manera dinámica e individual en cada uno de estos.

Todas las plataformas desarrolladas en Liferay funcionan bajo este paradigma, encapsulando las funcionalidades de la aplicación en distintos *Portlets*. Además, la plataforma y comunidad proveen de numerosos *Portlets* ya desarrollados, que permiten aprovechar el trabajo realizado por otros expertos en la plataforma y concentrarse en el desarrollo de aquellas aplicaciones o funcionalidades que no estén disponible.

Las **funcionalidades definidas para el presente proyecto tienen su implementación en un conjunto de Portlets**, tal y como se detalla en el próximo apartado.

- **Hook**. Se trata de una herramienta que provee Liferay que permite sobrescribir las funcionalidades propias del portal, facilitando las modificaciones para desarrolladores y administradores que deseen editar determinados parámetros o comportamiento del portal.
- ***Layout***. Se encarga de la distribución del contenido y aplicaciones dentro de una página del portal, permitiendo distintas configuraciones según la combinación de filas y columnas que desee el usuario.

  Es posible crear distintas plantillas de Layout para que una o más páginas se estructuren con el mismo diseño, aunque una página sólo puede configurarse con un Layout.

- ***Theme***. Se trata de una herramienta de Liferay encargada de gestionar la apariencia del portal web. La plataforma permite que el desarrollador del portal defina el aspecto y apariencia del portal, pudiendo crear un Theme

totalmente personalizado, o bien generar uno nuevo que se base en dos estilos predefinidos que ofrece la plataforma, *Styled* y *Unstyled*.

En referencia a las tecnologías que emplea Liferay para el desarrollo de los portales, uno de los aspectos más importantes, que hacen de esta plataforma uno de los sistemas de desarrollo de portales más extendidos y con mejor aceptación entre los profesionales, radica en el uso de estándares libres y generalizados para todas las funcionalidades del portal.

En este sentido, las tecnologías que con mayor recurrencia se emplean en el desarrollo son:

- **Java**, para el desarrollo de portlets y funcionalidades extendidas a las existentes en la plataforma base de Liferay. Para el desarrollo de portlets pueden emplearse dos frameworks distintos: Liferay Studio (versión para profesionales) y Eclipse + SDK plugin Liferay.
- ***JSP***, que permite desarrollar la interfaz gráfica generando contenido HTML o XML de manera dinámica. Para ello, esta tecnología emplea una máquina virtual de Java.
- **CSS**, en Liferay esta tecnología la provee la propia plataforma en su apartado de configuración para elaborar y estructurar la presentación estructura de documentos tipo HTML (por ejemplo, páginas web).
- **Javascript y Jquery**, que permite el desarrollo de funcionalidades y acceso a elementos HTML que se interpretan en el navegador (client-side), permitiendo otorgar de mayor dinamismo al usuario, así como distintos efectos visuales.

Finalmente, con su componente *Liferay CMS*, ofrece funcionalidades de adición, edición y eliminación de contenidos, mediante herramientas parametrizables y configurables como:

- **Publicación Web**. Incorpora un sistema completo de gestión de contenido web, que permite generar páginas ya elaboradas, emplear plantillas predefinidas, programar o retirar la publicación de contenidos.
- **Repositorio único de documentos y archivos multimedia**. La plataforma ofrece un punto de almacenamiento de contenidos, como ahora documentos y archivos multimedia.

- **Publicador de Contenidos**. Como CMS, la plataforma permite mediante un publicador de contenidos añadir y configurar acciones para mostrar información, publicada en el portal.
- **Editores avanzados de texto**. Ofrece herramientas propias para la edición de texto para la publicación de contenidos, incluyendo funcionalidades como corrector ortográfico, definición de estilos

Capítulo 3. Implementación de la Plataforma

En este capítulo se procede a ofrecer al lector del detalle de los principales aspectos relevantes en cuanto al desarrollo de la solución del presente proyecto, como ahora el entorno de trabajo, los portlets desarrollados y el despliegue del prototipo para su validación.

3.1 Entorno de trabajo

Para la fase de desarrollo de la plataforma se ha configurado un entorno de trabajo que permita la codificación y despliegue del prototipo del proyecto. En este sentido, el entorno de trabajo ha estado compuesto por los siguientes elementos:

- **IDE Eclipse Neon**. Para la codificación de los portlets de la plataforma se ha empleado el framework de desarrollo Eclipse, versión Neon, herramienta de código abierto multiplataforma, con arquitectura basada en RCP (Rich Client Platform) para el desarrollo en el lenguaje de programación Java y debugging de la solución con los gadgets disponibles (integración con Ant, Apache, etc.).
- **Liferay IDE plugin**. Este plugin integrable en la plataforma Eclipse incluye un SDK que permite agilizar el desarrollo de plugins Liferay como ahora portlets, themes, servicios, etc., permitiendo generar estructuras y código reutilizable de manera que el desarrollador pueda focalizarse en la codificación de la lógica de las aplicaciones.
- **Liferay Portal CE (Community Edition) 6.2 bundle with tomcat**. Se trata de la versión de código libre de la plataforma web empresarial de Liferay para crear soluciones empresariales que ofrecen resultados inmediatos. Permite el despliegue de portales desarrollados en un motor o servidor web Apache Tomcat.
- **PostgreSQL Server**. Se trata de un sistema de gestión de base de datos relacional orientada a objetos, de código abierto. Permite el almacenamiento de los datos del portal.
- **Virtual Box + Ubuntu *distro***. Se ha configurado una máquina virtual mediante la herramienta virtual box, que permita disponer de un despliegue portable para la comunicación entre el estudiante y el tutor del TFG, así como la

validación y entrega del producto final. Esta máquina se ha configurado con el sistema operativo de código abierto Ubuntu (GNU/Linux), la instalación de PostgreSQL Server y Liferay Portal CE 6.2.

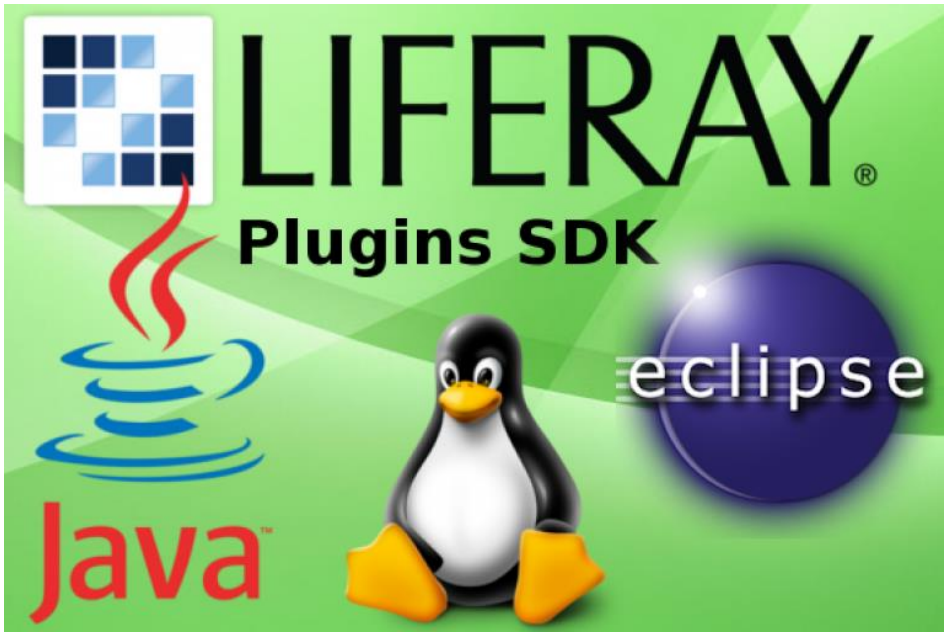


Ilustración 18 Principales tecnologías para el desarrollo y despliegue de la plataforma.

Para obtener un mayor detalle de la configuración del entorno de trabajo y despliegue de la plataforma, véase el Anexo 3 Despliegue del entorno.

3.2 Metodología Desarrollo

Tal y como se ha comentado en el apartado anterior, el entorno de trabajo seleccionado para el desarrollo del portal web permite al desarrollador centrarse principalmente en la lógica de la aplicación y olvidarse de otros aspectos que el propio *framework* que se emplea en el entorno de trabajo configura y estructura de manera autónoma.

En este sentido, para el desarrollo del portal **FLOSS .NET**, se ha empleado el Liferay IDE de Eclipse, que permite generar la estructura de clases necesarias para el portal, basándose en utilidades y plugins disponibles en el *framework*.

En específico para generar la estructura del portal, en primer lugar, se ha empleado el *framework* **Liferay MVC Portlet**, que permite el desarrollo de Portlets siguiendo los estándares JSR 168 y 226 y desplegarlos en la plataforma Liferay.

Este *framework* está diseñado y desarrollado específicamente para *Liferay* y dispone de múltiples características y APIs para el desarrollo y despliegue ágil de un portal basado en esta plataforma. Mediante su uso, no es necesario realizar configuraciones adicionales ni añadir archivos ".jar" durante el desarrollo, permitiendo la creación y despliegue de Liferay MVC Portlets (si se empleara el *framework* Generic Portlet, sería necesario configurar y desarrollar numerosos aspectos).

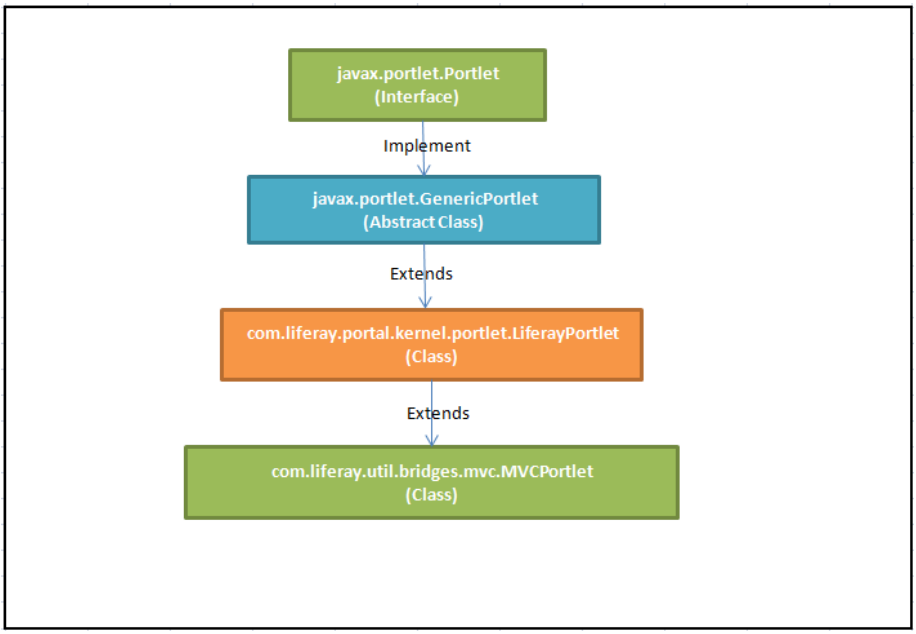


Ilustración 19 Jerarquía de portlets MVC.

Tal y como su nombre indica, el *framework* emplea el patrón de diseño Modelo-Vista-Controlador (MVC). Este patrón provee una manera relativamente simple para estructurar la aplicación del portal, separando las responsabilidades en distintos componentes.

Con este patrón de diseño, el **Controlador** es el encargado de la comunicación con la capa de presentación y la capa de modelo de la aplicación, recibiendo las peticiones que realizan los usuarios a través de la interfaz gráfica de la capa de presentación, procesando estas peticiones y enviando respuestas (request/response) a la capa de

persistencia o modelo. Es el encargado de determinar que acciones se ejecutan según las peticiones recibidas, procesar las acciones que actualizan la capa modelo y seleccionar la vista necesaria a mostrar una vez realizada la acción. Tiene su representación en clases Java.

La capa de **Presentación** contiene toda la lógica para mostrar los datos al usuario, permitiendo mostrar componentes HTML en función de las necesidades o propósitos definidos. Tiene su representación en archivos JSP.

La capa de **Modelo** contiene los datos de la aplicación y la lógica o reglas de negocio necesarias para la manipulación de estos datos. El *framework* dispone de un plugin, denominado Service Builder, que permite, mediante la definición y descripción en XML de las entidades y atributos que deberá contener la plataforma, generar la estructura completa de clases Java encargadas de modelar y manipular los datos en la BBDD, materializando de manera autónoma las sentencias SQL necesarias para la creación de tablas, edición, etc.

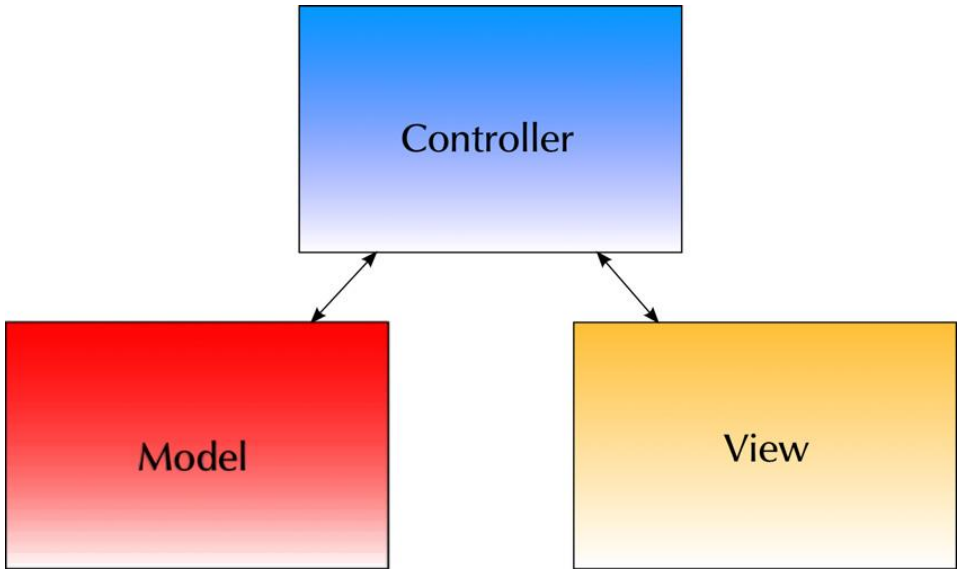


Ilustración 20 Patrón de diseño MVC.

De este modo, para el desarrollo del portal se ha generado un nuevo proyecto del tipo Liferay Plugin Project MVC y se ha usado el plugin *Service Builder*, que han permitido generar la estructura de clases necesarias para el desarrollo de la plataforma.

3.2 Desarrollo

3.2.1 Introducción

Tal y como se ha detallado en el portal anterior, en el inicio del desarrollo del portal se ha generado la estructura de clases necesarias mediante los *plugins* que ofrece el *framework*.

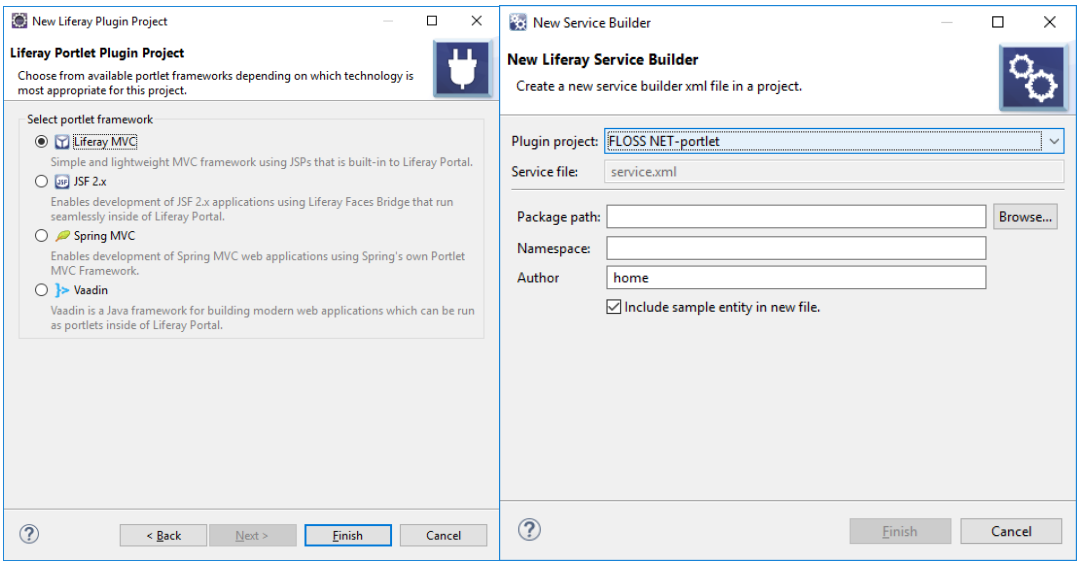


Ilustración 21 Creación de Liferay Plugin Project y Service Builder. Liferay MVC Portlet.

service.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE service-builder PUBLIC "-//Liferay//DTD Service Builder 6.2.0//EN" "http://www.liferay
<service-builder package-path="com.floss">
    <author>home</author>
    <namespace>sb</namespace>
    <entity name="Applicant" local-service="true" remote-service="false">
        <column name="id" primary="true" type="long"></column>
        <column name="userId" type="long"></column>
        <column name="professionalEnterprise" type="String"></column>
        <column name="telephone" type="String"></column>
        <column name="email" type="String"></column>
        <column name="reviewRank" type="double"></column>
        <column name="companyId" type="long"></column>
        <column name="groupId" type="long"></column>
        <order by="asc">
        <order-column name="id" />
        </order>
        <finder name="applicantFinder" return-type="Collection">
        <finder-column name="id" />
        </finder>
        <finder name="GroupId" return-type="Collection">
            <finder-column name="groupId" />
        </finder>
        <finder name="CompanyId" return-type="Collection">
            <finder-column name="companyId" />
        </finder>
        <finder name="UserId" return-type="Collection">
            <finder-column name="userId" />
        </finder>
```

Ilustración 22 Fragmento archivo XML codificado para ejecutar el servicio Service Builder.

El resultado de la creación de un nuevo proyecto, del tipo Liferay Plugin**, Liferay Portlet MVC** ha generado la siguiente estructura de carpetas:

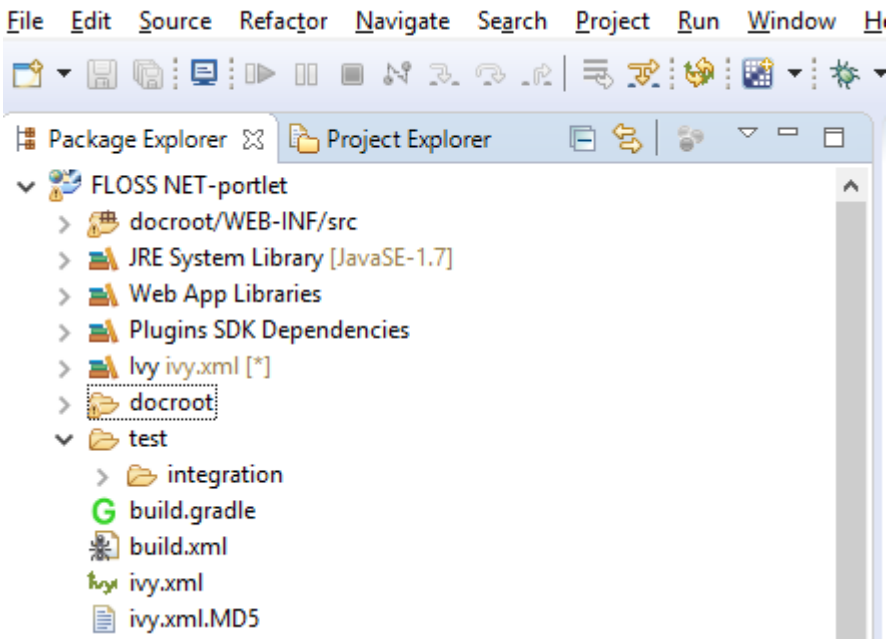


Ilustración 23 Estructura carpeta Liferay MVC Portlet.

Por otro lado, el resultado de ejecutar el servicio Service Builder, con el XML desarrollado, ha generado la siguiente estructura de carpetas.

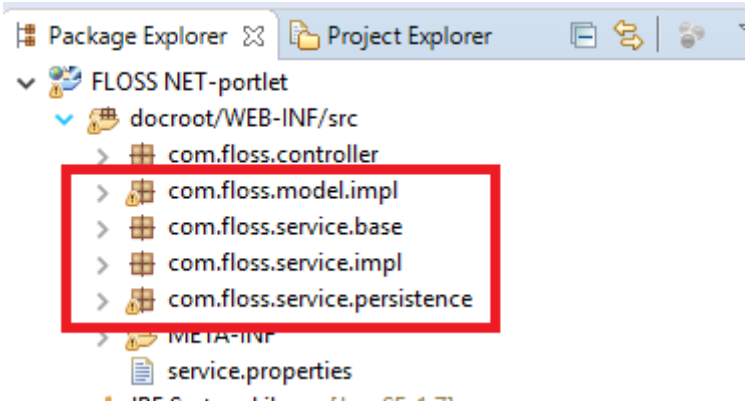


Ilustración 24 Paquetes de clases generados con Service Builder.

De este modo, el *framework* ha generado la estructura completa de carpetas, así como configuraciones necesarias para la codificación y desarrollo del portal, permitiendo

concentrase únicamente en el desarrollo de la lógica de la capa de presentación, negocio y manipulación de datos.

3.2.2 Controladores

En relación a la capa de negocio de la solución, se han generado un conjunto de clases Java que representan los controladores del portal.

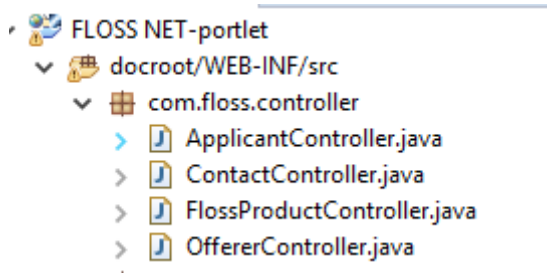


Ilustración 25 Clases controladoras del portal.

Se han creado 4 controladores, encargados de gestionar distintos aspectos:

- **ApplicantController**. Este controlador es el encargado de gestionar todas las operaciones los usuarios demandantes del portal. Se han codificado un conjunto de métodos que implementan las funcionalidades necesarias. Los métodos elaborados para este controlador han sido:
  - registerAsApplicant. Este método es el encargado de dar de alta a un usuario como demandante en la plataforma. Mediante los datos que el usuario aporta en un formulario en la capa de presentación, se comunica con la capa de persistencia para generar un registro en la BBDD almacenando la información del demandante y permite al usuario visualizar las funcionalidades propias de este rol.
  - getOfferers. Este método es el encargado de listar a petición del demandante los usuarios ofertantes disponibles en la BBDD del portal. Recibe la petición de la capa de presentación y se comunica con la capa de persistencia para obtener los ofertantes registrados. Después selecciona la vista de presentación de ofertantes y le suministra la información a la capa de presentación para mostrarlos.

- contactOfferer. Este método es el encargado de enviar solicitudes de contactos a usuarios ofertantes. Comunica la petición de contacto a la capa de persistencia, la cual devuelve si el envío ha sido satisfactorio o no, de manera que pueda retornar el resultado a la capa de presentación.
- deleteApplicant. Este método es el encargado de dar de baja al usuario como demandante. Recibe la petición del usuario por medio de la capa de presentación y comunica la intención de baja a la capa modelo. Esta a su vez comunica el éxito o no de la baja al controlador, que se encarga de retornar la respuesta a la capa de presentación para que muestre el resultado por pantalla.
- updateApplicant. Este método es el encargado de actualizar la información de un usuario demandante. Recibe los datos introducidos por un usuario demandante, traslada la petición a la capa de persistencia para que se efectúen y devuelve la respuesta de la transacción realizada por la capa modelo a la capa de presentación.
- createProject. Este método es el encargado de registrar un proyecto relacionado con una tecnología FLOSS para el usuario demandante. Recibe la petición y datos de registros de la capa de presentación, procesa la información y la traslada a la capa de persistencia para que esta cree el proyecto. Recibe la respuesta y devuelve el resultado a la capa de presentación.
- viewApplicant. Este método recibe una petición de un usuario ofertante que desea ver la información completa de un usuario demandante con el que haya contactado. Comunica la petición a la capa de persistencia y recibe los datos de esta para trasladarlos a la capa de presentación.

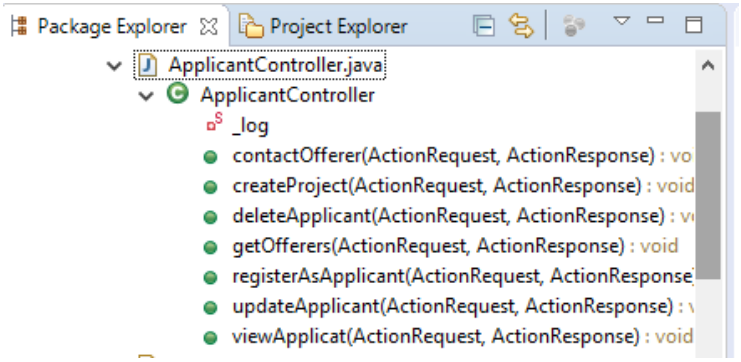


Ilustración 26 Métodos controlador demandantes.

- **OffererController**. Este controlador es el encargado de gestionar todas las operaciones los usuarios ofertantes del portal. Se han codificado un conjunto de métodos que implementan las funcionalidades necesarias. Los métodos elaborados para este controlador han sido:
    - registerAsOfferer. Este método es el encargado de dar de alta a un usuario como ofertante en la plataforma. Mediante los datos que el usuario aporta en un formulario en la capa de presentación, se comunica con la capa de persistencia para generar un registro en la BBDD almacenando la información del demandante y permite al usuario visualizar las funcionalidades propias de este rol.
    - getApplicants. Este método es el encargado de listar a petición del ofertante los usuarios demandantes disponibles en la BBDD del portal. Recibe la petición de la capa de presentación y se comunica con la capa de persistencia para obtener los demandantes registrados. Después selecciona la vista de presentación de demandantes y suministra la información a la capa de presentación para mostrarlos.
    - contactApplicant. Este método es el encargado de enviar solicitudes de contactos a usuarios demandantes. Comunica la petición de contacto a la capa de persistencia, la cual devuelve si el envío ha sido satisfactorio o no, de manera que pueda retornar el resultado a la capa de presentación.
    - deleteOfferer. Este método es el encargado de dar de baja al usuario como ofertante. Recibe la petición del usuario por medio de la capa de presentación y comunica la intención de baja a la capa modelo. Esta a su vez comunica el éxito o no de la baja al controlador, que se encarga de retornar la respuesta a la capa de presentación para que muestre el resultado por pantalla.
    - updateOfferer. Este método es el encargado de actualizar la información de un usuario ofertante. Recibe los datos introducidos por un usuario ofertante, traslada la petición a la capa de persistencia para que se efectúen y devuelve la respuesta de la transacción realizada por la capa modelo a la capa de presentación.
    - createCompetence. Este método es el encargado de registrar una competencia relacionada con una tecnología FLOSS para el usuario ofertante. Recibe la petición y datos de registros de la capa de presentación, procesa la información y la traslada a la capa de

persistencia para que esta genere la competencia. Recibe la respuesta y devuelve el resultado a la capa de presentación.

- viewOfferer. Este método recibe una petición de un usuario demandante que desea ver la información completa de un usuario ofertante con el que haya contactado. Comunica la petición a la capa de persistencia y recibe los datos de esta para trasladarlos a la capa de presentación.

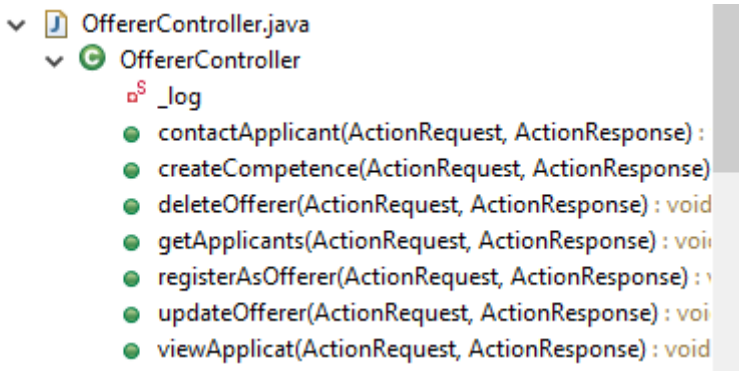


Ilustración 27 Métodos controlador ofertantes.

- **ContactController**. Este controlador es el encargado de gestionar todas las operaciones de contacto entre los usuarios del portal. Se han codificado un conjunto de métodos que implementan las funcionalidades necesarias. Los métodos elaborados para este controlador han sido:
  - acceptContact. Este método es el encargado de aceptar una petición de contacto que ha recibido un usuario. Recibe la petición de la capa de presentación y la comunica a la capa de persistencia. La capa modelo devuelve el resultado de la transacción en la BBDD y comunica a la capa de presentación el mismo.
  - applicantRequest. Este método es el encargado de listar las solicitudes de contacto de demandantes que ha recibido el usuario. Recibe la petición de la capa de presentación, comunica la misma a la capa de persistencia, la cual devuelve un listado con las peticiones que no hayan sido aceptadas previamente. El controlador devuelve el listado a la capa de presentación para que lo muestre.
  - offererRequests. Este método es el encargado de listar las solicitudes de contacto de ofertantes que ha recibido el usuario. Recibe la petición de la capa de presentación, comunica la misma a la capa de persistencia, la

cual devuelve un listado con las peticiones que no hayan sido aceptadas previamente. El controlador devuelve el listado a la capa de presentación para que lo muestre.

- declineContact. Este método es el encargado de rechazar una petición de contacto que ha recibido un usuario. Recibe la petición de la capa de presentación y la comunica a la capa de persistencia. La capa modelo devuelve el resultado de la transacción en la BBDD y comunica a la capa de presentación el mismo.

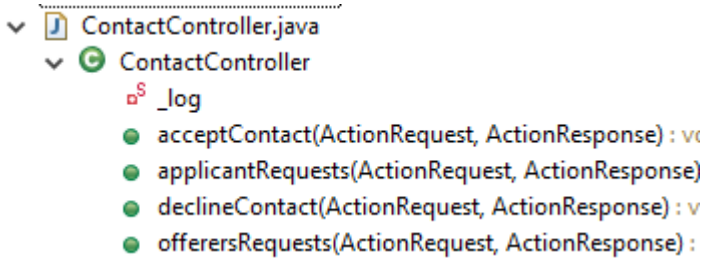


Ilustración 28 Métodos controlador de contactos.

- **FlossProductController**. Este controlador es el encargado de gestionar todas las operaciones que se realicen sobre el catálogo de productos Floss del portal. Se han codificado un conjunto de métodos que implementan las funcionalidades necesarias. Los métodos elaborados para este controlador han sido:
  - deleteFlossProduct. Método encargado de eliminar un producto floss del catálogo. Recibe la petición de un usuario administrador mediante la capa de presentación, comunica la misma a la capa de persistencia y devuelve la respuesta generada por esta a la vista con el resultado de la operación.
  - registerFlossProduct. Método encargado de registrar un producto floss del catálogo. Recibe la petición de un usuario administrador mediante la capa de presentación, comunica la misma a la capa de persistencia y devuelve la respuesta generada por esta a la vista con el resultado de la operación.
  - updateFlossProduct. Método encargado de actualizar un producto floss del catálogo. Recibe la petición de un usuario administrador mediante la capa de presentación, comunica la misma a la capa de persistencia y devuelve la respuesta generada por esta a la vista con el resultado de la operación.

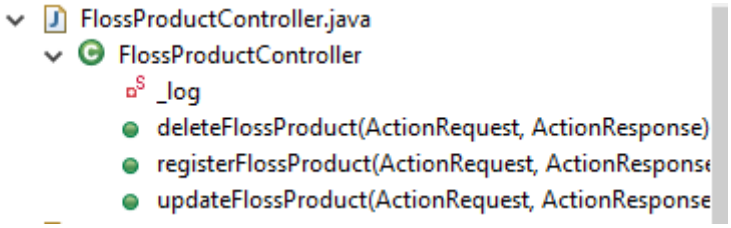


Ilustración 29 Métodos controlador de catálogo de productos FLOSS.

```
package com.floss.controller;

import java.io.IOException;

public class FlossProductController extends MVCPortlet {

private static Log _log = LogFactoryUtil.getLog(FlossProductController.class);

    @ProcessAction( name = "registerFlossProduct" )
    public void registerFlossProduct( ActionRequest req, ActionResponse res ) throws IOExceptio
        User user = (User) req.getAttribute(WebKeys.USER);

        String name = req.getParameter("name");
        String release = req.getParameter("release");
        String appType = req.getParameter("appType");
        String language = req.getParameter("language");
        String dataBases = req.getParameter("dataBases");
        long groupId = user.getGroupId();
        long companyId = user.getCompanyId();

        FlossProduct floss = FlossProductLocalServiceUtil.registerFloss(name,release,appType, l
        if (floss != null) {
        _log.info("Floss have been register successfylly");
        }
        res.setRenderParameter( "mvcPath", "/html/flossProductController/view.jsp" );
    }

    @ProcessAction( name = "deleteFlossProduct" )
    public void deleteFlossProduct(ActionRequest req, ActionResponse res) throws IOException, P
```

Ilustración 30 Ejemplo método controlador codificado. Controlador FlossProductController, método registerFlossProduct.

3.2.3 Capa modelo

En relación a la capa modelo o de persistencia de la solución, el servicio *Service Builder* ha generado la estructura, conjunto de clases y código que permite recibir las peticiones de la capa de negocio o controladores, trasladar la petición a la BBDD mediante consultas SQL (visualización o modificación/registro/eliminación de datos), y devolver la información y el resultado al controlador en cuestión.

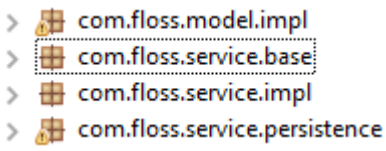


Ilustración 31 Paquetes con clases Java generados por el servicio.

En este sentido, cada uno de los paquetes tiene el siguiente propósito:

- **com.floss.model.impl**. Este paquete tiene como finalidad la de extender el paquete de persistencia, implementando un contendor que almacena la información proporcionada por la BBDD o el portal, para la ejecución de los distintos servicios y operaciones del resto de paquetes de la capa modelo.
- **com.floss.service.base**. Este paquete provee del servicio local necesario para la implementación e interfaz de las clases y métodos para el acceso, modificación, registro o eliminación de un registro en la tabla equivalente a una entidad del portal.
- **com.floss.service.persistence.** Este paquete tiene como finalidad modelar las entidades del portal, mediante clases y métodos de acceso, registro, modificación y eliminación de registros de las tablas relacionadas.
- **com.floss.service.impl.** Este paquete de clases java tiene como finalidad proveer del servicio necesario para implementar los métodos e interfaces de acceso, registro, modificación o eliminación. Se trata del único paquete que debe modificar el desarrollador ya que, cada vez que se incluyen métodos en una clase de este paquete, debe ejecutarse nuevamente el servicio *Service Builder* y genera los métodos o clases relacionados en el resto de paquetes de la capa de persistencia. Se han codificado tantos métodos por entidad como operaciones deben realizarse sobre esta (operaciones que ejecuta el controlador). En este sentido, las clases desarrolladas han sido:
    - **ApplicantLocalServiceImpl**. En esta clase se ha codificado toda la lógica de negocio para el acceso/modificación/eliminación de la tabla de BBDD que almacena la información de los demandantes.
    - **CompetenceLocalServiceImpl**. En esta clase se ha codificado toda la lógica de negocio para el acceso/modificación/eliminación de la tabla de BBDD que almacena la información de las competencias de ofertantes.
    - **ContactLocalServiceImpl**. En esta clase se ha codificado toda la lógica de negocio para el acceso/modificación/eliminación de la tabla de BBDD que almacena la información de los contactos del portal.

- **FlossProductLocalServiceImpl**. En esta clase se ha codificado toda la lógica de negocio para el acceso/modificación/eliminación de la tabla de BBDD que almacena la información del catálogo de productos FLOSS.
- **OffererLocalServiceImpl**. En esta clase se ha codificado toda la lógica de negocio para el acceso/modificación/eliminación de la tabla de BBDD que almacena la información de los ofertantes.
- **ProjectLocalServiceImpl**. En esta clase se ha codificado toda la lógica de negocio para el acceso/modificación/eliminación de la tabla de BBDD que almacena la información de los proyectos registrados por demandantes.

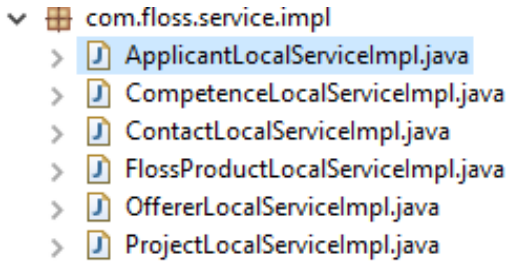


Ilustración 32 Clases java codificadas.

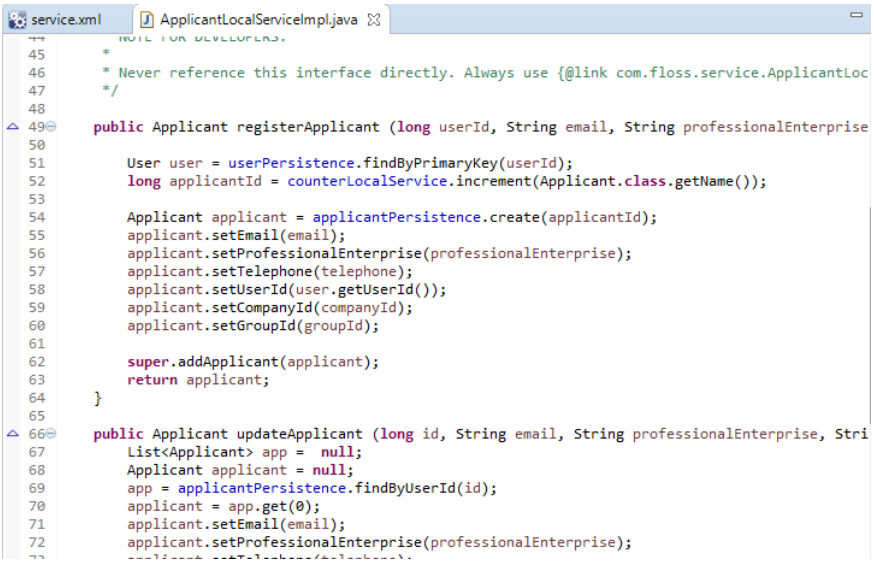


Ilustración 33 Ejemplo codificación método codificado para el registro de demandante en la BBDD.

3.2.3 Capa Presentación

En relación a la capa de presentación de la solución, está representada por archivos JSP codificados con el código HTML que genera las páginas de la aplicación. La capa de negocio selecciona la vista a mostrar al usuario y recibe las peticiones y envía la información de respuesta a la capa de presentación para mostrar los resultados al usuario.

Los archivos JSP codificados se encuentran almacenados en una estructura de carpetas que almacena todos los JSPs a los que tiene que tener acceso cada uno de los controladores del portal.

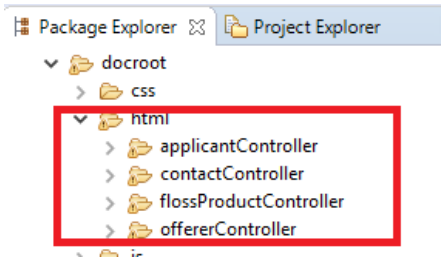


Ilustración 34 JSPs distribuidos por controlador.

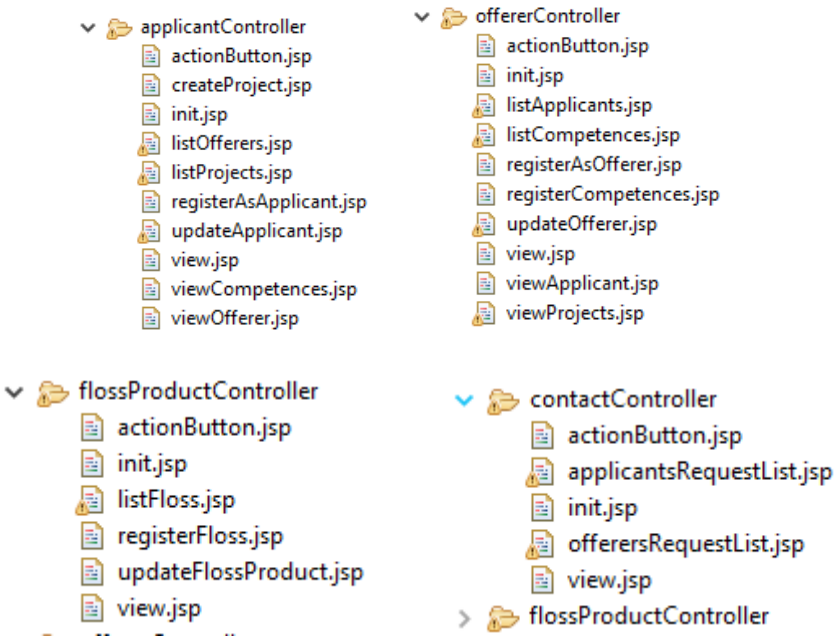


Ilustración 35 JSPs codificados.

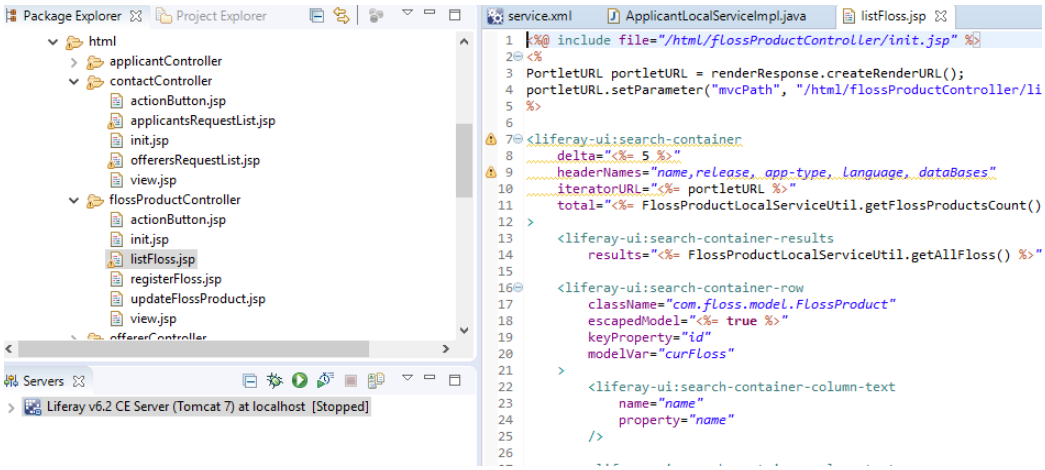


Ilustración 36 JSP codificado para mostrar el listado de productos floss del catálogo.

Cada uno de los controladores dispone de los distintos JSPs a los cuales accede y renderiza, de manera que los usuarios visualicen las páginas y contenido necesario en la aplicación web. Estos archivos, además de disponer del código HTML, contienen la codificación de operaciones java para la transformación y preparación de datos a mostrar, así como procesar la información necesaria que envían o reciben del controlador en cuestión.

```
35 <portlet:renderURL var="listOfferersURL">
36 <portlet:param name="mvcPath" value="/html/applicantController/listOfferers.jsp" />
37 </portlet:renderURL>
38
39 <portlet:renderURL var="createProjectURL">
40 <portlet:param name="mvcPath" value="/html/applicantController/createProject.jsp" />
41 </portlet:renderURL>
42
43 <portlet:renderURL var="listProjectsURL">
44 <portlet:param name="mvcPath" value="/html/applicantController/listProjects.jsp" />
45 </portlet:renderURL>
46
47 <portlet:actionURL var="deleteApplicantURL" name="deleteApplicant"/>
48
49
50 <p><a href="<%= listOfferersURL %>">List Offerers</a></p>
51
52 <%
53     if (exist) {
54 %>
55 <p><a href="<%=updateApplicantURL %>">Update Applicant data</a></p>
56 <p><a href="<%= deleteApplicantURL %>">Unregister as Applicant</a></p>
57 <p><a href="<%= createProjectURL %>">Create Project</a></p>
58 <p><a href="<%= listProjectsURL %>">List Applicant Projects</a></p>
59
60 <% }
61 else {%>
62
```

Ilustración 37 Código java condicional que muestra distintas opciones en función de si el usuario se encuentra dado de alta como demandante.

Además, como se ha podido observar en anteriores ilustraciones cada una de las estructuras de carpetas dispone de un archivo JSP, “view.jsp”. Este archivo contiene el código HTML que renderizará de inicio el controlador o portlet en cuestión, una vez el usuario lo haya añadido al mismo.

Ilustración 38 View.jsp del catálogo de productos FLOSS.

Finalmente, indicar que las peticiones y respuestas a los usuarios, se generan en el JSP mediante el uso de operaciones del tipo “render” o “action”.

Las acciones del tipo render permiten navegar entre los distintos JSP disponibles para el controlador (en el ejemplo anterior, view.jsp habilita enlaces para acceder a formulario de registro de productos floss y el listado de productos).

Las acciones del tipo action envían una petición al controlador, el cual se encarga de gestionarla y devolver los datos necesarios a mostrar por la capa de presentación.

3.2.3 Registro de Portlets

Como se ha comentado anteriormente, las aplicaciones desarrolladas con la plataforma *Liferay* son en sí mismas un contenedor de Portlets, en el cual el portal es un Portlet que contiene otros Portlets.

Además de los desarrollos codificados y descritos a lo largo de la presente memoria, para que la plataforma pueda desplegar los Portlets desarrollados es necesario el registro de estos.

Este registro se realiza mediante la definición del Portlet, aportando la información relativa a este y el mapeo con los JSPs a los cuales tendrá acceso en un archivo XML denominado "Portlet.xml". El archivo puede modificarse de manera directa o mediante el plugin que ofrece el IDE.

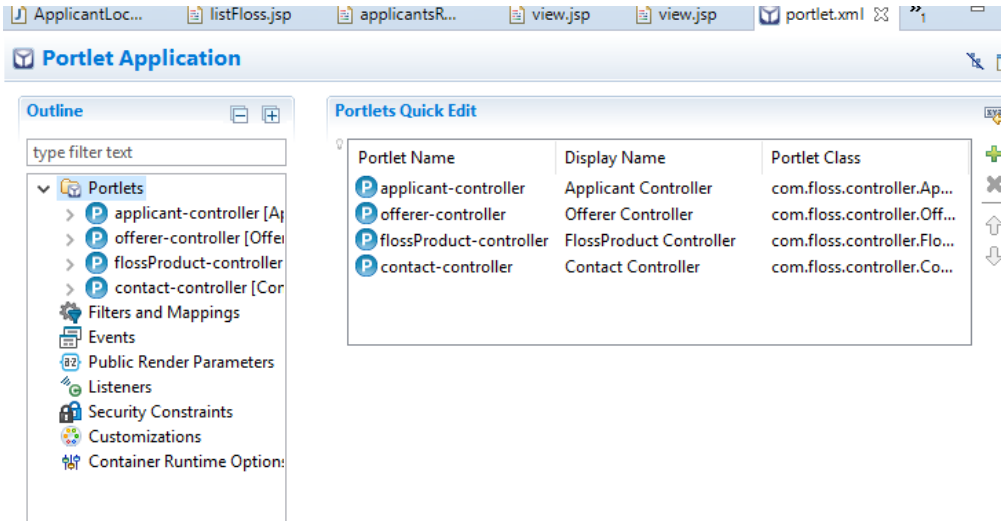


Ilustración 39 Portlets definidos. Plugin.

```
1  <?xml version="1.0"?>
2
3  <portlet-app xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd" xmlns:xsi="http://ww
4
5      <portlet>
6          <portlet-name>applicant-controller</portlet-name>
7          <display-name>Applicant Controller</display-name>
8          <portlet-class>
9              com.floss.controller.ApplicantController
10         </portlet-class>
11         <init-param>
12             <name>view-template</name>
13             <value>/html/applicantController/view.jsp</value>
14         </init-param>
15         <expiration-cache>0</expiration-cache>
16         <supports>
17             <mime-type>text/html</mime-type>
18             <portlet-mode>view</portlet-mode>
19         </supports>
20         <portlet-info>
21             <title>Applicant Controller</title>
22             <short-title>Applicant Controller</short-title>
23             <keywords></keywords>
24         </portlet-info>
25         <security-role-ref>
26             <role-name>administrator</role-name>
27         </security-role-ref>
28         <security-role-ref>
29             <role-name>guest</role-name>
```

Ilustración 40 Edición Portlet.xml. Definición ApplicantController.

Como puede observarse en el XML, el acceso a la vista por defecto en el caso de Applicant Controller se define mediante la ruta “/html/applicantController/view.jsp”, y a partir de ese punto el propio JSP será el encargado de recibir las peticiones y mostrar respuestas a los usuarios.

Finalmente, para que la plataforma registre y muestre los Portlets codificados, es necesario definir en qué categoría se encuentran, en el archivo “liferay-display.xml”, indicando el nombre de la categoría y asociando el id del controlador registrado en el archivo Portlet.xml.

listFloss.jsp | applicantsR... | view.jsp | view.jsp | portlet.xml | liferay-dis...

```
1  <?xml version="1.0"?>
2  <!DOCTYPE display PUBLIC "-//Liferay//DTD Display 6.2.0//EN" "http://www.liferay.com/dtd
3
4  <display>
5
6      <category name="category.sample">
7          <portlet id="applicant-controller"></portlet>
8          <portlet id="offerer-controller"></portlet>
9          <portlet id="flossProduct-controller"></portlet>
10         <portlet id="contact-controller"></portlet>
11     </category>
12 </display>
```

Ilustración 41 Archivo Liferay-display.xml del portal.

Para el proyecto se han registrado todos los portlets en la categoría de muestras (sample), ya que el propósito del mismo es la construcción de un prototipo y no la materialización de un producto final.

Capítulo 4. Resultados Obtenidos

4.1 Introducción

El objetivo de este apartado reside en el análisis y presentación de los resultados obtenidos con la ejecución del proyecto. En este sentido, se pretende ofrecer el detalle de los Portlets desarrollados.

Tal y como se ha comentado anteriormente, una de las ventajas que tiene el uso de la tecnología Liferay para la implementación del portal es que permite a los usuarios configurar a su conveniencia las aplicaciones (Portlets) en su página de inicio.

En el caso del presente proyecto, se ha configurado la solución mediante las herramientas disponibles en la plataforma, de manera que al acceder al portal mediante la función *Login*, en función de su rol el usuario pueda acceder a distintas páginas privadas para las cuales se ha dispuesto los controladores desarrollados (no obstante, el usuario tiene disponible la opción de añadir o quitar los controladores a los cuales tenga acceso).

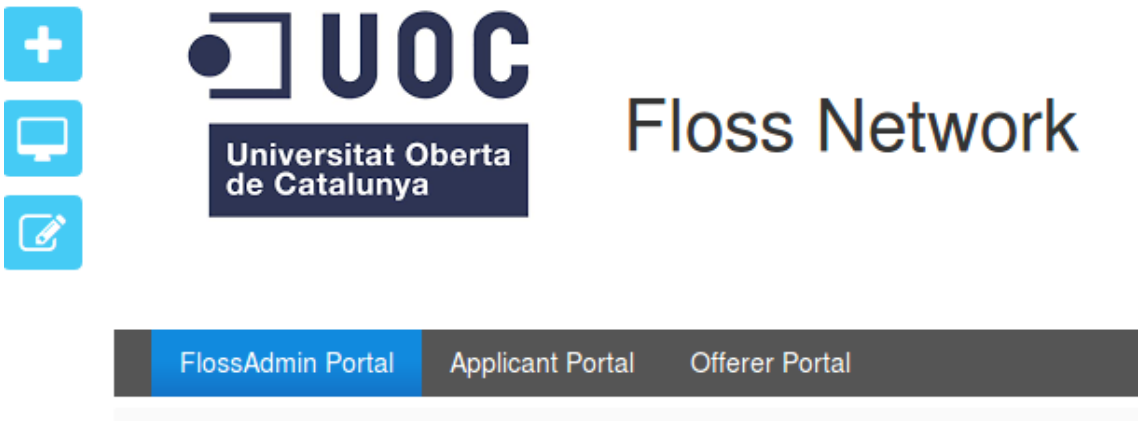


Ilustración 42 Menú páginas privadas del portal. Acceso como usuario administrador.

Las páginas privadas que se han configurado son:

- **FlossAdmin Portal**. Esta página contiene el controlador FlossProductController, con las operaciones de gestión del catálogo de

productos floss del portal. Su acceso está configurado únicamente para usuarios del tipo administrador.

- **Applicant Portal**. Esta página contiene el controlador ApplicantController, con las operaciones disponibles relacionadas con usuarios demandantes o usuarios que quieran darse de alta como tal. Asimismo, contiene el controlador ContactController, que permite la gestión de solicitudes de contacto recibidas para el usuario.
- **Offerer Portal**. Esta página contiene el controlador OffererController, con las operaciones disponibles relacionadas con usuarios ofertantes o usuarios que quieran darse de alta como tal. Asimismo, contiene el controlador ContactController, que permite la gestión de solicitudes de contacto recibidas para el usuario.

4.2 FlossAdmin Portal

La página FlossAdmin Portal, contiene el controlador FlossProductController. Las opciones que puede realizar el usuario administrador mediante el acceso a la página son:

- **Registrar productos Floss**. Pulsando en el enlace se accede al formulario de registro de un nuevo producto al catálogo FLOSS del portal. Deben cumplimentarse los datos del formulario y pulsar el botón guardar para realizar el registro.
- **Listar tecnologías Floss registradas**. Pulsando el enlace se accede al listado de tecnologías Floss del catálogo, con la información asociada a cada una de estas y un botón de acciones que permite al administrador del portal editar o eliminar la tecnología Floss en cuestión.

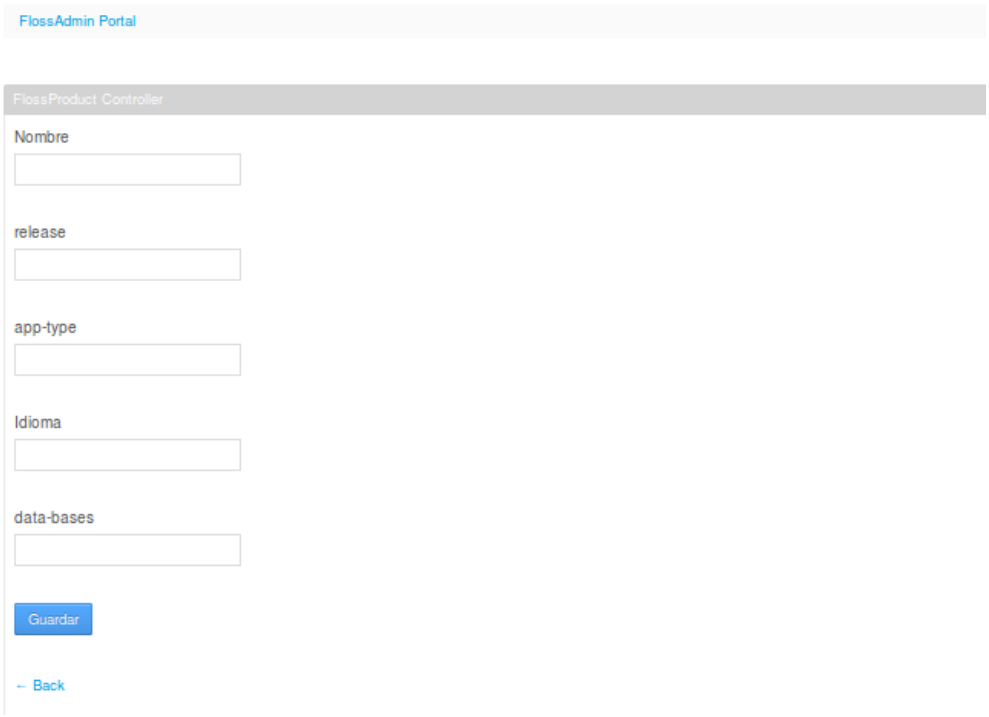


Ilustración 43 Registro de producto Floss.

Ilustración 44 Listado de tecnologías FLOSS del catálogo.

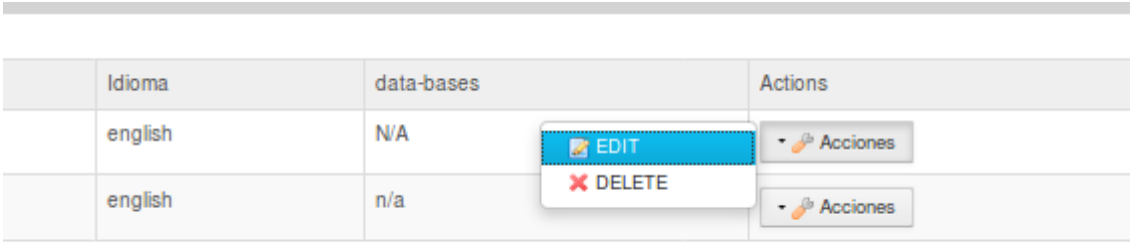


Ilustración 45 Acciones disponibles para tecnología GNU/Linux del catálogo.

4.3 Applicant Portal

La página Applicant Portal, contiene el controlador ApplicantController y el controlador ContactController. Las opciones que puede realizar el usuario mediante el acceso a la página son:

- **Listar usuarios ofertantes.** Pulsando el enlace se accede al listado de usuarios ofertantes de la plataforma, con la información básica asociada a cada uno de estos y un botón de acciones que permite al usuario enviar una solicitud de contacto al ofertante y visualizar los datos de contacto. Si se selecciona visualizar los datos de contacto, el usuario accede a una nueva página en la que tendrá las siguientes opciones disponibles:
    - **Visualizar las competencias del ofertante**.
    - **Puntuar y comentar la colaboración con el ofertante**.
    - **Visualizar la puntuación media del ofertante.**

- **Darse de alta/baja como demandante.** En función de si el usuario está dado de alta o no como demandante, visualizará un enlace u otro en la página. Además, si no se encuentra dado de alta como tal, únicamente tendrá disponible en la página Applicant Portal la posibilidad de listar ofertantes o registrarse como demandante.
- **Crear un proyecto.** Pulsando el enlace se accede a un formulario para el registro de un proyecto sobre una tecnología del catálogo de productos FLOSS.
- **Listar los proyectos creados por el usuario.** Pulsando el enlace se accede a un listado con los proyectos que ha registrado el usuario como demandante, con la información completa de estos.
- **Actualizar sus datos como demandante.** Pulsando el enlace se accede a un formulario con los datos registrados del demandante, en el cual es posible modificar los mismos y actualizarlos.
- **Visualizar solicitudes de contacto enviadas por otros usuarios.** Pulsando el enlace se accede a un listado de solicitudes de contacto recibidas por el usuario, con la posibilidad de aceptar y generar el contacto, o bien o rechazar estas.

Ilustración 46 Listado de opciones disponibles para un usuario registrado como demandante.

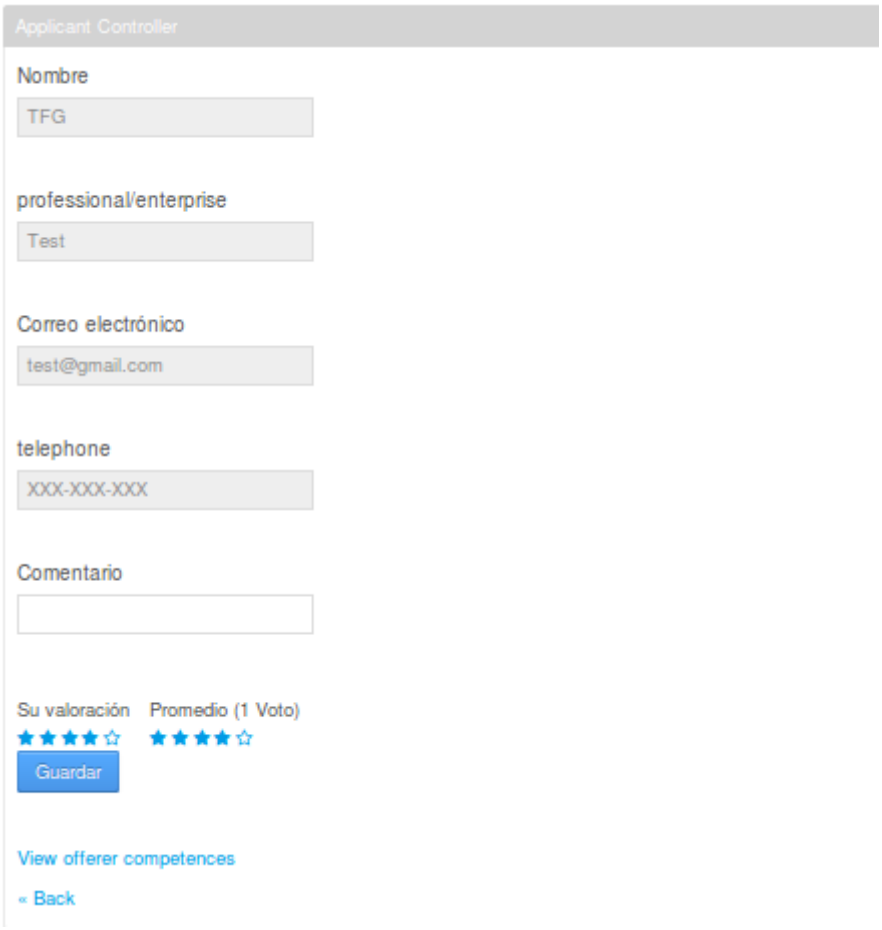


Ilustración 47 Visualización de un ofertante. Acceso desde el listado de ofertantes.

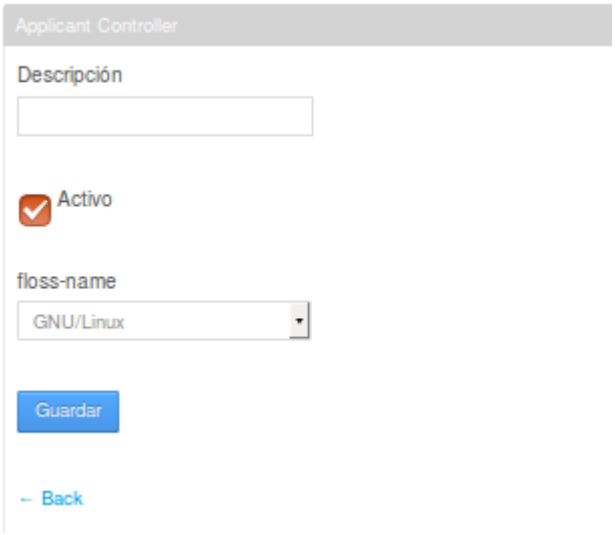


Ilustración 48 Registro de un nuevo proyecto para la tecnología FLOSS GNU/Linux del catálogo.

4.4 Offerer Portal

La página Offerer Portal, contiene el controlador OffererController y el controlador ContactController. Las opciones que puede realizar el usuario mediante el acceso a la página son:

- **Listar usuarios demandantes.** Pulsando el enlace se accede al listado de usuarios demandantes de la plataforma, con la información básica asociada a cada uno de estos y un botón de acciones que permite al usuario enviar una solicitud de contacto al demandante y visualizar los datos de contacto. Si se selecciona visualizar los datos de contacto, el usuario accede a una nueva página en la que tendrá las siguientes opciones disponibles:
  - **Visualizar los proyectos registrados del demandante**.
  - **Puntuar y comentar la colaboración con el demandante**.
  - **Visualizar la puntuación media del demandante.**

- **Darse de alta/baja como ofertante.** En función de si el usuario está dado de alta o no como ofertante, visualizará un enlace u otro en la página. Además, si no se encuentra dado de alta como tal, únicamente tendrá disponible en la

página Applicant Portal la posibilidad de listar demandantes o registrarse como ofertante.

- **Registrar una competencia.** Pulsando el enlace se accede a un formulario para el registro de una competencia sobre una tecnología del catálogo de productos FLOSS.
- **Listar competencias registradas por el usuario.** Pulsando el enlace se accede a un listado con las competencias que ha registrado el usuario como ofertante, con la información completa de estas.
- **Actualizar sus datos como ofertante.** Pulsando el enlace se accede a un formulario con los datos registrados del ofertante, en el cual es posible modificar los mismos y actualizarlos.
- **Visualizar solicitudes de contacto enviadas por otros usuarios.** Pulsando el enlace se accede a un listado de solicitudes de contacto recibidas por el usuario, con la posibilidad de aceptar y generar el contacto, o bien o rechazar estas.

Ilustración 49 Listado de opciones disponibles para un usuario registrado como ofertante.

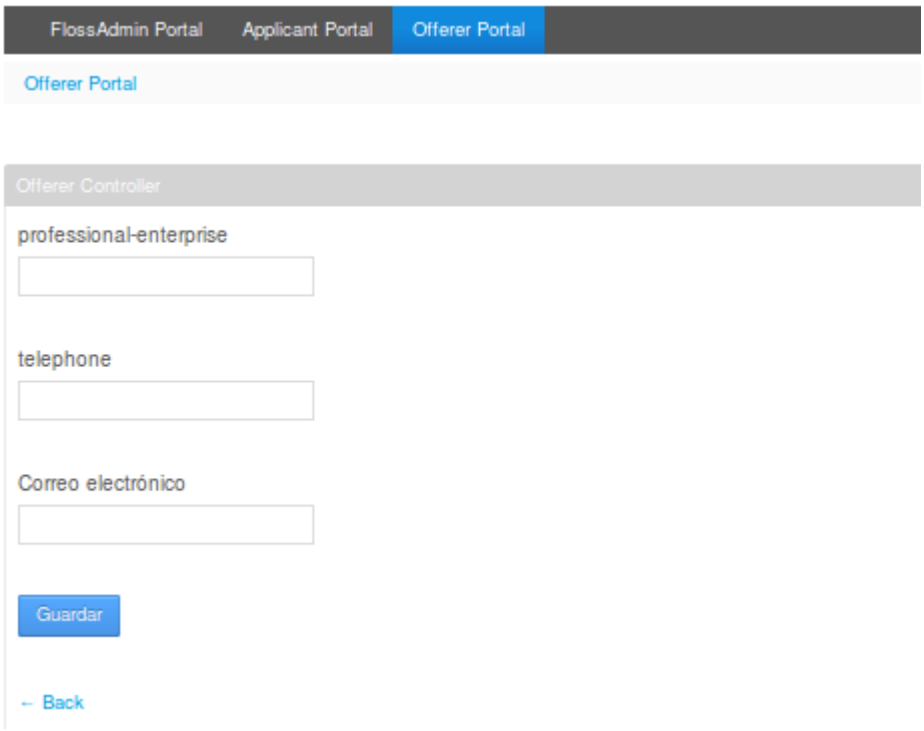


Ilustración 50 Actualización de datos de ofertante.

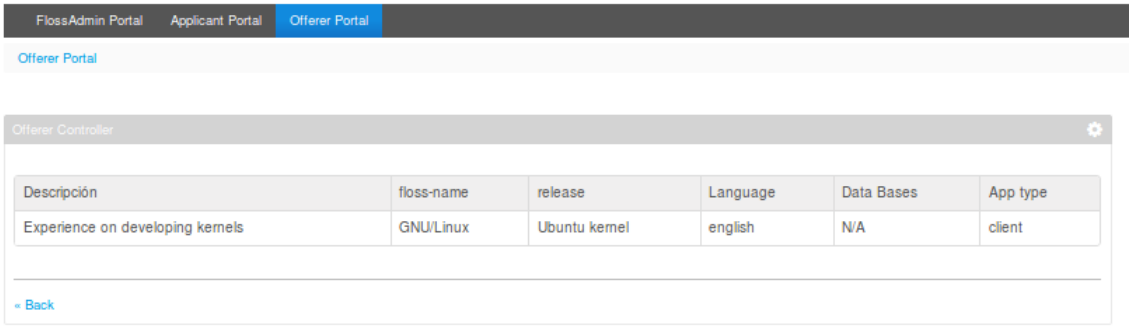


Ilustración 51 Visualización de competencias registradas para un ofertante.

Capítulo 5. Conclusiones

En este apartado se lleva a cabo el análisis y valoración de la tecnología *Liferay*, elaborando las conclusiones del trabajo realizado. A este respecto, se resumen los principales pasos llevados a cabo para la obtención de la solución a la problemática planteada, recopilando los aspectos más relevantes.

8.1 Conclusiones del proyecto

A raíz del trabajo realizado es posible demostrar que la aplicación de una metodología organizada y planificada en fases para el desarrollo de soluciones o sistemas software permite asegurar el éxito del trabajo, generando una solución con la calidad mínima para su demostración.

El objetivo principal del trabajo residía en el desarrollo de un portal web que permitiera el acceso a distintos tipos de usuarios, demandantes y ofertantes de proyectos de tecnologías FLOSS, para establecer un punto de encuentro que les facilite el primer contacto para iniciar una colaboración.

A este respecto, el portal desarrollado satisface con éxito este objetivo principal y prácticamente la totalidad de los sub-objetivos planteados. No obstante, se habían definido algunos sub-objetivos opcionales, que son la gamificación de la solución y la elaboración de un sistema de insignias de reputación según la valoración realizada entre usuarios para las colaboraciones realizadas, que no han podido cumplirse y han quedado fuera del alcance del proyecto por motivos de recursos (no se disponía de suficiente tiempo para su desarrollo).

Asimismo, la motivación y trasfondo del proyecto busca generar una plataforma que permita difundir y compartir el conocimiento con el fin de crear una comunidad de usuarios que puedan conectarse y establecer contactos profesionales, generando un nuevo modelo de negocio inexistente hasta el momento que tiene como filosofía propagar y extender el uso de tecnologías FLOSS a empresas que no dispongan de recursos suficientes para acometer los proyectos.

En referencia a la planificación, he podido comprobar la dificultad que conlleva la estimación de costes y recursos para la ejecución de tareas relacionadas con un proyecto, y la importancia de mantener el compromiso a nivel de tiempo y calidad de

los resultados de estas tareas para garantizar el éxito. En este sentido se ha intentado seguir en la medida de lo posible la planificación elaborada en la primera fase, sin embargo, debido a distintas dificultades encontradas en la ejecución de las tareas, ha sido necesario dedicar mayor esfuerzo para mantener la planificación.

Así, durante la fase de diseño pude comprobar que los diseños inicialmente planteados o que se comprendían de manera clara una vez realizado el análisis de la solución, difícilmente pueden mantenerse si no se es experto en la tecnología que se va a emplear en el proyecto, ya que influyen de manera significativa en su evolución y versión final aplicada.

En la misma línea, los desarrollos que inicialmente se plantean sencillos, ya que la propia plataforma *Liferay* propone soluciones y grandes funcionalidades desarrolladas por la comunicad y disponibles en las APIs, pueden verse contrariados por las propias limitaciones que plantea la tecnología como configuraciones no soportadas, o *bugs* detectados que no tienen solución en la línea temporal del proyecto.

No obstante, durante esta misma fase de desarrollo, es necesario destacar las lecciones aprendidas en el uso de una tecnología, desconocida hasta el inicio del proyecto por el autor, que ha permitido el desarrollo de una solución de manera ágil y en un tiempo acotado, que por la contra sin el uso de *frameworks* o plataformas similares su alcance se habría visto afectado.

De este modo, se ha podido corroborar como el uso de una plataforma como *Liferay* permite optimizar los recursos y esfuerzos dedicados, y concentrarlos en aquellos aspectos más relevantes y funcionales de una aplicación, revelándose como una poderosa herramienta a tener en consideración en futuros proyectos que permite diseñar y desarrollo soluciones colaborativas atractivas y que podrían tener una implantación relativamente sencilla en un entorno productivo.

8.2 Líneas de trabajo futuras

A partir de la solución obtenida en el proyecto se ha materializado un prototipo de la solución a la problemática inicialmente planteada. No obstante, se trata de una versión que dista de poder ser implantada en un entorno real y que sería necesario aplicar un

proceso de rediseño y codificación para su uso productivo, tal y como se plantea en el Anexo 2 Plan de Implantación.

Bibliografía

[1] *Liferay* Inc, *Liferay* portal: [on line] *Liferay*, Inc., 2017 [Consulta: 30 de marzo 2017], Disponible en: < https://www.liferay.com/es/solutions/portals>
[2] *Joomla*: About *Joomla*! [on line]. *Open Source Matters, Inc*., 2017 [Consulta: 30 de marzo 2017] Disponible en:< https://www.joomla.org/about-joomla.html>
[3] *Drupal*: About *Drupal* [on line]. *Dries Buytaert,* 2017 [Consulta: 30 de marzo 2017] Disponible en:< https://www.drupal.org/about >
[4] *Liferay* Inc, *Liferay* Portal 6.2 - Development Guide [on line]. *Liferay*, Inc. ,2017 [Consulta: 15 de abril de 2017], Disponible en:
< https://dev.liferay.com/es/develop/tutorials/-/knowledge_base/6-2/tutorials >
[5] *Portlets in Action* [edición digital] *Sarin, Ashish, 2011. Manning* [Consulta:18 de abril de 2017], Disponible en:
< http://pdf.th7.cn/down/files/1312/Portlets%20in%20Action.pdf >
[6] *Liferay in Action* [edición digital] *Richard Sezov, Jr, 2011. Manning* [Consulta:20 de abril de 2017], Disponible en:
< http://serwis.kampus.uj.edu.pl/cds/books/Liferay.in.Action.pdf >

Glosario

CMS

Sistema de Gestión de Contenidos, que permite crear una estructura de soporte (framework) para la creación y administración de contenidos, principalmente de páginas web.

Java

Lenguaje de programación orientado a objetos que permite la codificación y ejecución de aplicaciones en tiempo de ejecución.

HTML

Lenguaje de Marcas de Hipertexto, que permite la descripción de estructuras y contenido en formato textual, complementando el texto con imágenes o archivos multimedia.

JSP

Java Server Pages. Tecnología que permite el desarrollo de páginas web con programación java.

XML

Lenguaje de Marcas Extensibles cuyo funcionamiento se basa en el uso de etiquetas. Se trata de un estándar para el intercambio de información estructurada entre plataformas.

Framework

Entorno de trabajo con conceptos, prácticas o herramientas estandarizadas que permite enfocar un problema empleando una referencia para resolverlo.

SDK

Kit de desarrollo de software, compuesto por un conjunto de herramientas que permite a los desarrolladores crear aplicaciones para un sistema en concreto.

Portal

Página inicial web que permite la navegación por esta misma mediante el acceso del usuario.

Portlet

Componentes modulares de interface de usuario que son gestionados y visualizados en un portal web.

MVC

Modelo Vista Controlador, es un patrón de diseño de arquitectura software que separa en capas los datos, la lógica de negocio y la interfaz del usuario.

Plugin

Aplicación informática que se integra o relaciona con otra aplicación para agregarle una función nueva y generalmente específica.

Índice de Anexos

ANEXO 1

Informe de seguimiento 1

Portal de comunicación software libre (proveedores <->pequeña/micro-empresa)

Enrique Hidalgo Ayllón

15/03/2017

Dirección: Javier Martí Pintanel

Tabla de contenido

Ejecución de Fases del proyecto

FASE 1 Planteamiento del proyecto

La FASE 1 del proyecto ha consistido en una toma de contacto con el mismo. A partir de la idea inicial que se había concebido a la hora de elegir el área que atañe al TFG, se ha procedido a definir y plantear formalmente el proyecto con la colaboración del director, en cuanto a:

- Contexto.
- Alcancel del proyecto.
- Objetivos.
- Materias implicadas.
- Estado del Arte.
- Motivación personal.
- Conocimiento y/o experiencia en el entorno del problema.
- Líneas de desarrollo futuras.
- Medios disponibles.
- Enfoque y método.

El detalle de la definición del proyecto puede observarse en el entregable correspondiente a esta fase, **Planteamiento del proyecto**, adjunto a este informe.

Finalmente remarcar que esta fase ha concluido con éxito.

FASE 2 Planificación y desarrollo

Se ha iniciado la FASE 2 del proyecto, con la ejecución de las actividades de Planificación del proyecto. En este sentido, se han establecido:

- Hitos del proyecto.
- Elaboración de una planificación temporal desglosada en EDTs y tareas.
- Estimación de costes del proyecto.
- Análisis de riesgo del proyecto.

El detalle de los resultados obtenidos en la Planificación del proyecto puede observase en el entregable correspondiente, **Planificación del proyecto**, adjunto a este informe.

Remarcar que la ejecución de esta fase no ha finalizado y únicamente se han ejecutado actividades relativas al EDT Planificación, que con la elaboración del entregable mencionado queda cerrado.

Decisiones relevantes

En la ejecución de la FASE 1 del proyecto y las actividades de Planificación de la FASE 2 se han tomado las siguientes decisiones relevantes en relación al proyecto:

- **FASE 1**. En esta primera fase, en la definición del proyecto se había planteado enfocar el alcance del mismo en la implementación de un portal web de contacto entre ofertantes y demandantes, para pequeñas y micro-empresas, de implantaciones de ERP OpenSource. No obstante, esta idea quedó descartada al tratarse de un sector muy pequeño de herramientas, puesto que solo existen 2 o 3 sistemas de esta tipología. En este sentido, se amplió el alcance del proyecto a una plataforma web de contacto entre ofertantes y demandantes (pequeña y micro-empresas) de implantaciones de las plataformas, sistemas y soluciones existentes para el mundo de la empresa bajo el paradigma del software libre.
- **FASE 2**. No han existido decisiones relevantes hasta el momento en la ejecución de esta fase. Remarcar los hitos principales definidos, detallados en el documento de planificación del proyecto.

Informe de seguimiento 2

FLOSS NET. *Free/Libre and Open Source Software Network*

Enrique Hidalgo Ayllón

12/04/2017

Dirección: Javier Martí Pintanel

Tabla de contenido

FASE 2 Planificación y desarrollo

Se continúa con la ejecución de la FASE 2 del proyecto, con las **actividades de Análisis** del proyecto, relativas al EDT Análisis. En este sentido se han establecido y/obtenido los siguientes resultados:

- Se ha elaborado una definición del sistema a alto nivel.
- Se han identificado y formalizado los requisitos funcionales y no funcionales de la solución.
- Se han identificado y analizado los principales ítems del sistema y funcionalidades asociadas.
- Se han analizado plataformas tecnológicas del sector y se ha **identificado la tecnología** a emplear, **cumpliendo con uno de los hitos establecidos en el proyecto**.

Por otro lado, se han iniciado las **actividades de Diseño**, relativa al EDT Diseño. En este sentido, se han establecido y/obtenido los siguientes resultados:

- Se ha obtenido un diagrama de casos de uso general con las distintas funcionalidades y/o operaciones para cada actor del sistema.
- Se ha obtenido un primer diagrama a alto nivel de las principales entidades de la plataforma.

Remarcar que la ejecución de esta fase no ha finalizado. Queda pendiente la ejecución y finalización de actividades de diseño de la plataforma, como ahora el diseño del portal y la BBDD de la solución.

Decisiones relevantes

En la ejecución de la FASE 2 EDT de análisis se han tomado las siguientes decisiones relevantes en relación al proyecto:

- **FASE 2**. Una vez analizadas de manera minuciosa las tecnologías potenciales para la plataforma base, se ha identificado como plataforma base a emplear la tecnología Liferay. Por otro lado, se ha identificado y formalizado los requisitos del proyecto y se han valorado los mismos según la definición del alcance del proyecto y los conocimientos disponibles, asignando niveles de criticidad. Finalmente se han identificado los principales componentes de la solución y se han analizado, sentando las bases técnicas para la ejecución del EDT de análisis.

Informe de seguimiento 3

FLOSS NET. *Free/Libre and Open Source Software Network*

Enrique Hidalgo Ayllón

31/05/2017

Dirección: Javier Martí Pintanel

Tabla de contenido

ESTADO DEL PROYECTO

El proyecto ha mantenido ha cumplido con la planificación realizada a nivel de ejecución de actividades en el tiempo. No obstante, para la PAC3 estaba planificada la generación de un entregable definitivo con diseño presentado en la fase, pero durante la fase de implementación se ha rediseñado la solución, por lo que el entregable no ha sido generado.

Las desviaciones y modificaciones en el diseño son debidas al desconocimiento de la tecnología escogida, ya que existían características propias de la plataforma base (en específico gestión de usuarios), que estaban planteadas en el desarrollo, por lo que ha sido necesario el rediseño durante la implementación.

Por otro lado, ha sido necesario dedicar grandes esfuerzos para la implementación de la solución debido a:

- En primer lugar, desconocimiento de la tecnología. Este hecho ha supuesto una importante dedicación de tiempo durante la fase de implementación de diversas fuentes para la inmersión en las características, potencial y posibilidades de Liferay, así como las distintas librerías y estructuras mediante tutoriales para llevar a cabo el desarrollo.
- En segundo lugar, inicialmente se optó por el desarrollo en un entorno LIFERAY IDE, pero con la plataforma CE 7 (última versión estable). A pesar de la estabilidad de la solución, el uso de esta versión comportó grandes problemas por incompatibilidades para el desarrollo de plug-ins, así como problemas en la configuración con el SDK de java, ya que para la versión JDK 8 no era posible el uso de ciertos componentes básicos y necesarios.

Para la configuración del entorno de implementación durante esta fase ha habido grandes complicaciones puesto que inicialmente se trabajó con una versión que no permitía la ejecución correcta. Este punto se detalla en el apartado de desviaciones.

En referencia al seguimiento con el tutor del proyecto, durante esta etapa ha existido mayor comunicación estableciendo dos llamadas para comentar con el tutor complicaciones en la configuración del entorno y pautas para el desarrollo, además de comunicados por parte del tutor mediante el fórum, para aclaraciones al respecto.

Finalmente, indicar que las actividades planificadas se han ejecutado al 100%, resultando en un entregable mínimo como prototipo de la plataforma. Queda pendiente la documentación del rediseño realizado, que se aportará documentado en el apartado de DISEÑO de la Memoria del TFG.

ENTREGABLES ADJUNTOS

Junto al Informe de Seguimiento, se adjuntan los siguientes documentos:

- Máquina Virtual OVA con la configuración e instalación del software para la ejecución del prototipo.
- Archivos .war con el *deployment* de los Portlets desarrollados para la ejecución en el entorno.
- Documento Anexo con los pasos para la ejecución y despliegue de la plataforma (se incluye como archivo README en el escritorio de la MV).
- Documento de plan de implantación con los principales puntos a tener en cuenta.

RIESGOS Y DESVIACIONES

Tal y como se ha comentado en el apartado de estado de ejecución, en la presente etapa han surgido distintas cuestiones que han afectado a la planificación del proyecto.

En primer lugar, durante la preparación y configuración del entorno de programación, han existido problemas significativos en la integración de las herramientas a emplear, así como incompatibilidad de versiones. A pesar de ser una plataforma con un grado de madurez relevante, la última versión estable disponible Liferay CE 7 presenta grandes problemas de compatibilidad con la versión Java empleada, así con la integración con el entorno Liferay IDE de eclipse. Inicialmente se apostó por esta configuración *Liferay CE 7 bundle with Tomcat + Liferay IDE (Eclipse) + Open JDK8 + HSQL* (base de datos por defecto de la plataforma, con una interfaz de gráfica de administración poco amigable), no obstante después de distintas pruebas y errores encontrados con el entorno, se ha empleado la configuración Liferay CE 6.2 GA5 bundle with tomcat + Liferay IDE (eclipse) + Open JDK7 + PostgreSQL, instaladas en la máquina virtual para subsanarlo.

En segundo lugar, se inició el desarrollo de los Portlets siguiendo el diseño planteado en la etapa anterior y empleando una estructura comentada previamente con el tutor por capas:

- Persistencia: clases modelo y DAO.
- Servicios: clases Controller con la lógica de negocio y el mapeo entre las vistas y los procedimientos.
- JSP: Clases con el código HTML con la implementación de las vistas.

En referencia al **diseño**, se había especificado una clase general para Usuario, que extenderían las subclases *Applicant* y *Offerer*. Liferay dispone de su propio Portlet de Login/Registro, con su propia clase modelo User, y si se empleaba una clase desarrollada propia era necesaria la implementación de un *Hook* del login, además de la creación de esta clase. Por ello se ha considerado modificar el diseño y emplear como clase User, la propia que provee el sistema. En la misma línea se había planteado que el contacto entre *Offerers* y *Applicants* se realizara en función de un proyecto en

específico. Esta funcionalidad ha sido modificada, planteando el contacto independientemente de la existencia de un proyecto, de manera que en aquellos casos en los que se realice vigilancia o se desee plantear una posible colaboración para un proyecto no definido, pueda realizarse.

Las cuestiones destacadas, han impactado de manera sustancial en la ejecución del proyecto, siendo necesario para paliar las desviaciones una mayor dedicación e implicación para conseguir el éxito del proyecto.

Decisiones relevantes

En la ejecución de la FASE 2, actividades del EDT Implementación/Desarrollo, se han tomado las siguientes decisiones relevantes en relación al proyecto:

- Cambio de entorno de desarrollo y despliegue del prototipo. Finalmente se emplea Liferay 6.2 GA5 bundle with tomcat+ Liferay IDE (eclipse) + PostgreSql.
- Rediseño y desarrollo de la solución.
    - Para la elaboración de las clases modelo, persistencia y servicios, se había planteado elaborar la estructura de clases de manera manual con la generación de una clase modelo y otra clase DAO para cada item del sistema. No obstante, durante el desarrollo, después de investigar las característias que ofrece el IDE de Liferay, se ha optado por generar la estructura de clases con Service Builder.
    - Se emplea clase modelo liferay.portal.model.User.
    - A pesar de usar un modelo de base de datos relacional (Postgresql), se implementa la solución elaborando el desarrollo de los Portlets, de manera que estos se encarguen de actualizar las tablas de bases de datos pertinentes (estas se aislan, en primer lugar porque la plataforma da errores al intentar mapear foreing keys, en segundo porque interesa mantener un históricos de proyectos y usuarios registrados como demandantes y ofertantes para poder explotar los datos históricos).
    - El flujo necesario para contactar a un usuario se modifica, eliminando la necesidad de contactar para un proyecto específico y permitir el contacto para usuarios Ofertantes/Demandantes con el propósito de visualizar sus datos de contacto para futuras colaboraciones.
    - Se implementan finalmente 4 clases controladoras, que extienden la clase de liferay MVCPortlet:
        - Applicant Controller.
        - Offerer Controller.
        - Contact Controller.
        - FlossProductController.
    - Aunque inicialmente se había previsto gestionar el acceso a los portlets empleando el sistema de grupos y roles que ofrece la plataforma, se emplea código Java en los JSP de las vistas para este propósito ya que permite el uso de menos controladores y que estos encapsulen las funcionalidades propias necesarias.

ANEXO 2

Plan de implantación

FLOSS NET. *Free/Libre and Open Source Software Network*

Enrique Hidalgo Ayllón

31/05/2017

Dirección: Javier Martí Pintanel

Tabla de contenido

Introducción

El presente documento tiene como objetivo constituir una breve guía esquemática con los principales aspectos a tener en cuenta en el proceso de implantación y puesta en marcha del sistema FLOSS. NET desarrollado en el presente proyecto.

De este modo, se establecen los pasos a seguir y demás aspectos que serían necesario contemplar a lo largo del proceso. En este plan se pueden identificar 4 áreas generales, que son:

- **Desarrollo de producto final**. El producto obtenido es un entregable mínimo o prototipo para el cual es necesario realizar distintos ajustes y desarrollos en el código elaborado.
- **Planificación de implantación**. Con la definición de las actividades necesarias para llevarlo a cabo.
- **Configuración.** Incluye las necesidades identificadas mínimas de configuración (de la plataforma, no el entorno) para el despliegue final del sistema.

Desarrollo del prototipo

Tal y como se ha indicado en la introducción, la versión entregada del producto es un prototipo para el cual sería necesario destinar recursos con el objetivo de obtener una versión madura y estable que pueda implantarse en un entorno real.

A continuación se listan y describen los principales aspectos identificados, para los cuales es necesario destinar los recursos de desarrollo:

- **Capa de negocio:**
  - Componentes Controller, paquete *com.floss.Controller* de la solución. Dado el margen de tiempo y las desviaciones sufridas detalladas en el Informe de Seguimiento 3, ha sido necesario el desarrollo ágil del código de estos componentes, por lo que sería necesario recodificar estos componentes, optimizando el código en cuestiones referentes a:
    - Validaciones. Se han implementado validaciones a alto nivel que permitan la ejecución de las funcionalidades, no obstante no se validan los datos de entrada que se almacenan en la BBDD, lo que podría impactar en fallos de acceso y seguridad del sistema.
    - Funcionalidades. Los accesos a clases desarrolladas para el servicio (XXXLocalServiceUtil) de cada una de las entidades de la plataforma, se realizan de manera directa en cada uno de los controladores. Esto provoca un fuerte acoplamiento del prototipo. Para mejorar el mantenimiento y desacoplar la solución, se podría implementar una clase general controladora que mapee las vistas y subcontroladores específicos para cada entidad, de manera que puedan añadirse nuevos controladores y entidades sin tener que modificar las llamadas en la capa de Presentación (JSP) y aumente el nivel de complejidad del desarrollo.
- **Capa de presentación**:
  - Theme. Liferay proporciona la capacidad de crear un Theme (o plantilla) para la presentación del portal. En este sentido, se ha empleado el Theme Styled de la plataforma, pero de cara a la implantación en una organización o caso real, sería necesario configurar este theme para agregar un aspecto visual más atractivo a la misma.
  - JSPs. Los JSP elaborados consisten en páginas que permitan mostrar los datos y presentar una interfaz a los usuarios para las peticiones a la capa de negocio de la plataforma. De cara a una implantación real, sería necesario modificarlos optimizando el código desarrollado (redundancias, mensajes de éxito/error de peticiones, validaciones de formularios).
- **Capa de persistencia:**
  - BBDD. La plataforma se encarga mediante la configuración de acceso de la BBDD, de generar las tablas necesarias para su funcionamiento, con los archivos SQL que se incluyen en el archivo.war del proyecto. No obstante, las relaciones de las tablas no

han sido elaboradas de manera que no han sido introducidas claves foráneas, ya que existen beneficios para mantener el histórico de proyectos (asociados a demandantes) y competencias (asociadas a ofertantes). No obstante, algunas cuestiones como la eliminación del FlossProduct, debería actualizar estas tablas, pero al intentar añadir claves foráneas al ServiceBuilder de la solución e investigando en distintos foros de soporte a la plataforma, se ha detectado que no es posible implementar de manera efectiva este tipo de relaciones. En el sistema final deberían codificarse las relaciones mediante código java de manera que el funcionamiento sea el deseado.

Planificación y Descripción de Fases

En este apartado se define y desglosa las actividades de manera esquemáticas las actividades necesarias, para la implantación del sistema en un caso real.

- **Planificación del proyecto**. En esta fase se realizarán las siguientes actividades según el ámbito en que se ejecute la implantación:
    - Asignación de director. En esta actividad se decidirá la figura encargada de la dirección de la implementación.
    - Asignación de recurso para la configuración del entorno y red (perfil sistemas). En esta actividad se decidirán los recursos necesarios para preparar el entorno de despliegue de la plataforma, como ahora preparar el servidor de aplicaciones, servidor de Base de Datos, recursos de red necesarios.
    - Asignación de jefe de proyecto. Actividad de asignación de recurso para la ejecución del plan, supervisando las actividades de desarrollo a medida necesarias (detalladas en el apartado anterior), así como el resultado final obtenido.
    - Asignación de recursos de desarrollo. Personal encargado del desarrollo y parametrización a medida del sistema para el caso real.
    - Elaboración de tiempo de ejecución del plan. Estimación de coste o recursos, así como la planificación temporal para la implantación del sistema final.
- **Acondicionamiento**. En esta fase se realizará el acondicionamiento del entorno de desarrollo, de validación y en la última etapa del proyecto de producción del sistema. Será necesario preparar el entorno para:
    - **Desarrollo de la solución**. Entorno destinado para el desarrollo a medida de la solución en el cual sea posible desplegar el proyecto con motivos del desarrollo. La configuración mínima para este entorno radicará en un sistema operativo con:
        - JDK versión 7 instalado.
        - Liferay CE 6.2 GA5, bundle with tomcat.
        - Liferay IDE (eclipse).
        - Navegador.
        - Servidor de BBDD con PostgreSQL.

    - **Validación**. Entorno que contendrá los resultados obtenidos en el desarrollo (ficheros .war), la plataforma, así como configuración del equipo para el acceso de los Test users, encargados de las validaciones.
    - **Producción**. Entorno final en el cual se implementará la solución con los recursos hardware y software escalados a las necesidades del caso real, así como la solución estable del producto.
- **Desarrollo a medida**. Tal y como se ha definido anteriormente, en esta fase se ejecutarán los desarrollos necesarios para obtener una solución estable y a medida para el caso real planteado. En este fase se ejecutarán las actividades en el entorno de desarrollo definido.
- **Validación**. Fase que se ejecutará en el entorno de validación en la cual se realizarán:
    - Validación del código y funcionalidades desarrolladas o modificadas.
    - Validación de los Usuarios para la aceptación del producto final.
- **Despliegue**. En esta fase se asignarán los recursos definitivos al sistema a implantar y se realizará la configuración y despliegue del sistema, ofreciendo a usuarios finales el acceso al mismo para su uso.

Configuración

La plataforma FLOSS. NET, una vez elaborados los desarrollos mencionados, deberá ser configurada en mediante la asignación de un **usuario administrador del portal**, que sea capaz de configurar el mismo otorgando roles y permisos de acceso a los usuarios, así como encargándose del mantenimiento de la BBDD de productos Floss mediante el uso del controlador FlossAdmin.

ANEXO 3

Para la validación y prueba de los Portlets desarrollados se ha dispuesto del siguiente software:

- Sistema Operativo: Maquina Virtual con Ubuntu 16.04 instalada.

  superusuario: tfg

  password: Tfg2017

- DB:

  Se ha instalado como Base de Datos Postgresql 9 para permitir almecenar los datos y establecer la persistencia de la plataforma. La base de datos por defecto de Liferay HSQL tiene un entorno para visualizar los datos poco amigable. Para la interacción se ha instalado el plugin Pgadmin3. Los parametros de conexión a la BBDD son:

  usuario: postgres

  password: postgres

  puerto: 5432

  DB: lportal

- Plataforma:

  Se ha instalado la plataforma Liferay Portal CE 6.2 GA5 budle with tomcat. Esta plataforma no tiene una instalación como tal, sino que se ejecuta el archivo /startup.sh para iniciar el servicio poder acceder a la plataforma vía web.

Puesta en marcha:

1 Para la puesta en marcha del servicio, acceder vía terminal a la siguiente ruta:

  cd /usr/liferay-portal-6.2-ce-ga5/tomcat-7.0.62/bin

2 En esta ruta, ejecutar el archivo startup mediante el siguiente comando:

  ./startup.sh

3 Esperar a que arranque el servidor. Para visualizar el momento de arranque podemos observar el log catalina.out y observar que la consola muestre el mensaje de inicio finalizado (puede tardar varios minutos). Ejecutar el siguiente comando para visualizar el log:

  tail -f /usr/liferay-portal-6.2-ce-ga5/tomcat-7.0.62/logs/catalina.out

4 Minimizar la consola y abrir una pestaña de Firefox. En la barra del navegador se debe indicar la siguiente web (puede tardar algunos segundos si el servicio ha sido inicializado recientemente):

"localhost:8080"

5 Esperar mientras el servidor da respuesta y se abre la página de inicio del portal. Para loguearse en el portal usar las siguientes credenciales:

usuario: "hidalgo.enrique22@gmail.com"

password: "Tfg2017"