

LortuKodea

Autor: Jesús Blasco Armiñana

Plan de Estudios del Estudiante: Sistemas de información

Área del trabajo final: Desarrollo Web

Nombre Consultor/a: Gregorio Robles Martínez

Nombre Profesor/a responsable de la asignatura: Santi Caballe Llobet

Fecha Entrega: 9 Enero de 2020



Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada [3.0 España de Creative Commons](https://creativecommons.org/licenses/by-nc-nd/3.0/es/)

Licencias alternativas (elegir alguna de las siguientes y sustituir la de la página anterior)

A) Creative Commons:



Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada [3.0 España de Creative Commons](#)



Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-CompartirIgual [3.0 España de Creative Commons](#)



Esta obra está sujeta a una licencia de Reconocimiento-NoComercial [3.0 España de Creative Commons](#)



Esta obra está sujeta a una licencia de Reconocimiento-SinObraDerivada [3.0 España de Creative Commons](#)



Esta obra está sujeta a una licencia de Reconocimiento-CompartirIgual [3.0 España de Creative Commons](#)



Esta obra está sujeta a una licencia de Reconocimiento [3.0 España de Creative Commons](#)

B) GNU Free Documentation License (GNU FDL)

Copyright © AÑO TU-NOMBRE.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free

Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

A copy of the license is included in the section entitled "GNU Free Documentation License".

C) Copyright

© (el autor/a)

Reservados todos los derechos. Está prohibido la reproducción total o parcial de esta obra por cualquier medio o procedimiento, comprendidos la impresión, la reprografía, el microfilme, el tratamiento informático o cualquier otro sistema, así como la distribución de ejemplares mediante alquiler y préstamo, sin la autorización escrita del autor o de los límites que autorice la Ley de Propiedad Intelectual.

FICHA DEL TRABAJO FINAL

Título del trabajo:	<i>LortuKodea</i>
Nombre del autor:	<i>Jesús Blasco Armiñana</i>
Nombre del consultor/a:	<i>Gregorio Robles Martínez</i>
Nombre del PRA:	<i>Nombre y dos apellidos</i>
Fecha de entrega (mm/aaaa):	01/2020
Titulación:::	<i>Sistemas de Información</i>
Área del Trabajo Final:	<i>Desarrollo Web</i>
Idioma del trabajo:	<i>Castellano</i>
Palabras clave	<i>Liferay, Desarrollo Web, plantillas</i>
<p>Resumen del Trabajo (máximo 250 palabras): <i>Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.</i></p>	
<p>El proyecto consiste en desarrollar un portal web, con la última versión de Liferay. El proyecto está enmarcado dentro de la necesidad de dar respuesta rápida a ciertas necesidades de tiempo y recursos para la construcción de páginas web en un primer momento estáticas.</p> <p>Por tanto, aprovechando los conocimientos técnicos adquiridos durante los últimos años en Liferay 6.2 se aprovecha todo el conocimiento obtenido para realizar un estudio de su última versión estable 7.2 y trabajar sobre ella para la generación de un CMS generador de plantillas.</p> <p>Teniendo como objetivo la generación de estructuras en código HTML. Por lo que, el sitio web deberá contener un control de roles de usuarios, un desarrollo para la edición/generación de plantillas, con las diferentes funcionalidades que implica. Destacar, inclusión de texto enriquecido con un apartado multimedia en la nube, la personalización del menú, agrupación en grupos de plantillas y la gestión de las mismas.</p>	
<p>Abstract (in English, 250 words or less):</p>	

Índice

1. INTRODUCCIÓN	1
1.1. CONTEXTO Y JUSTIFICACIÓN DEL TRABAJO	1
1.2. OBJETIVOS DEL TRABAJO	1
1.3. ENFOQUE Y MÉTODO SEGUIDO	2
1.4. PLANIFICACIÓN DEL TRABAJO	3
1.5. BREVE SUMARIO DE PRODUCTOS OBTENIDOS	4
1.6. BREVE DESCRIPCIÓN DE LOS OTROS CAPÍTULOS DE LA MEMORIA	5
2. ESTUDIO DE COMPETENCIA	6
2.2. VENTAJAS Y DESVENTAJAS	7
2.3. FORTALEZAS Y FUNCIONALIDADES	9
3. ANÁLISIS DE LA APLICACIÓN	10
3.1. REQUISITOS FUNCIONALES	10
3.2. ROLES	10
3.3. CONFIGURAR PERMISOS	11
3.4. CASOS DE USO	11
4. DISEÑO DE LA APLICACIÓN	14
4.1. ARQUITECTURA LIFERAY	14
4.2. TECNOLOGÍA Y PROGRAMAS A UTILIZAR	15
4.3. ARQUITECTURA DEL PORTAL	17
4.4. COMPONENTES DEL PORTAL	17
4.5. BASE DE DATOS	18
4.5.1.- Reestructuración del Service-Builder	21
5. IMPLEMENTACIÓN	23
5.1. INSTALACIÓN ENTORNO DE DESARROLLO	23
5.1.1. Creación de las páginas	25
5.2. DESARROLLO DEL SERVICE BUILDER	26
5.2.1. Creación y configuración	26
5.2.2. API	28
5.2.3. Despliegue	30
5.3. DESARROLLO DE PORTLETS (MODELO)	30
5.3.1. Creación y configuración	30
5.3.2. Estructura e implantación	32
5.3.3. Despliegue	32
5.4. FORMULARIO DE CONTACTO	32
5.5. FAQs	34
6. DESARROLLO	37
6.1. PORTLET: PLANTILLAS	37
6.1.1. Seleccionar y diseñar estructura	37
6.1.2. Añadir/Editar texto	41
6.1.3. Contenido multimedia en la nube	42
6.1.4. Añadir/Editar menú	43
▪ Paleta de colores	43
▪ Botones dinámicos	44
▪ Redes sociales	44

6.1.5. Crear grupo y guardar.....	45
6.1.6. Comprimir y descargar.....	46
6.2. PORTLET: VISOR DE PLANTILLAS	47
6.2.1 Acciones	49
6.3. INVITAR A MIEMBROS	49
6.4. BLOG DE NOTICIAS	50
7. MANUAL DE USUARIO	51
7.1. REGISTRO	51
7.2. CREA MÍ PRIMERA PLANTILLA.....	52
7.3. ADMINISTRAR MÍ PLANTILLA.....	52
8. CONCLUSIONES.....	53
✓ Primera fase:	53
✓ Segunda fase:	53
9. GLOSARIO	55
10. BIBLIOGRAFÍA	56
11. ANEXOS.....	57
ANEXO A: LIFERAY: CONCEPTOS BÁSICOS	57
ANEXO B: INSTALACIÓN Y CONFIGURACIÓN BBDD	60

Lista de figuras

Ilustración 1: Diagrama de Grantt.....	3
Ilustración 2: Primer caso de uso	11
Ilustración 3: Segundo caso de uso	13
Ilustración 4: Estrcutura de Liferay	15
Ilustración 5: Arquitectura de Liferay	17
Ilustración 6	23
Ilustración 7: Despliegue de Portlets, ServiceBuilders	24
Ilustración 8: Primera página del portal	24
Ilustración 9: Indicativo de que la instancia está iniciada correctamente	25
Ilustración 10: Creación de un espacio de trabajo.....	25
Ilustración 11: Estructuración importante del portal.....	25
Ilustración 12: Edición de la estructura básica	26
Ilustración 13: Creación del ServiceBuilder	27
Ilustración 14: Service.xml	27
Ilustración 15: Compilador del ServiceBuilder	28
Ilustración 16: Ficheros a modificar del ServiceBuilder por parte del usuario ..	28
Ilustración 17: Ejemplo de contenido en un LocalServiceImpl.java	29
Ilustración 18: Zona de pruebas de las peticiones a la API	29
Ilustración 19: Ejemplo de petición disponible en la API	30
Ilustración 20: Despliegue ServiceBuilder	30
Ilustración 21: Creación portlet.....	30
Ilustración 22: Configuración Portlet y referencias necesarias	31
Ilustración 23: Dependencias del Portlet	31
Ilustración 24: Dependencias de la parte Front	32
Ilustración 25: Proceso de creación Formulario de Contacto	33
Ilustración 26: Configuración Servidor de correo.....	34
Ilustración 27: Opciones del servidor de correo	34
Ilustración 28: Creación plantilla FAQs	35
Ilustración 29: Estructura de FAQs.....	35
Ilustración 30: Código fuente del estilo de FAQs.....	36
Ilustración 31: Como añadir FAQs al portal.....	36
Ilustración 32: Multiplicar plantilla automáticamente y añadir la pregunta y la respuesta.....	37
Ilustración 33: Resultado FAQs.....	37
Ilustración 34: Código de la petición al cargar la página principal del portlet Plantillas.....	37
Ilustración 35: Código página principal del portlet de Plantillas.....	38
Ilustración 36: Resultado página principal	38
Ilustración 37: Dispositivos móviles	38
Ilustración 38: Llamada en el proceso de votación/puntuación de plantillas	39
Ilustración 39: Llamada al hacer click sobre una plantilla principal	39
Ilustración 40: Código página de edición de plantillas	40
Ilustración 41: Resultado del código anterior.....	40
Ilustración 42: Ejemplo de resultado obtenido.....	41
Ilustración 43: Insertar texto enriquecido.....	41
Ilustración 44: Insertar contenido multimedia en la nube	41

Ilustración 45: Ejemplo de inserción de imagen en la nube	42
Ilustración 46: Edición de texto.....	42
Ilustración 47: Construcción del servicio de edición de texto	43
Ilustración 48: Paleta de Color del menú.....	43
Ilustración 49: Código fuente de la paleta de selección	43
Ilustración 50: CSS de la paleta de Color del Menú	43
Ilustración 51: Código fuente creación botones menú.....	44
Ilustración 52: Resultado botones Menú	44
Ilustración 53: Resultado	44
Ilustración 54: Código fuente redes sociales	45
Ilustración 55: Resultado ventana de añadir Menú	45
Ilustración 56: Ventana de guardado.....	45
Ilustración 57: Selección de grupo ya existente	45
Ilustración 58: Perfil.....	47
Ilustración 59: Visor de plantillas del usuario.....	48
Ilustración 60: Ejemplo de petición a la API del Visor de plantillas.....	48
Ilustración 61: Código fuente que carga el contenido del Visor de plantillas	48
Ilustración 62: Código del Visor de plantillas del Usuario	49
Ilustración 63: Código de los botones de acción	49
Ilustración 64: Notificaciones.....	49
Ilustración 65: Ventana invitar a miembros	50
Ilustración 66: Blog.....	50
Ilustración 67: Formulario de alta de usuarios.....	51
Ilustración 68: Acceso usuarios registrados	51
Ilustración 69: Ejemplo de selección de plantilla principal	52
Ilustración 70: Ejemplo de edición de estructura de plantilla	52

1. Introducción

1.1. Contexto y justificación del Trabajo

LortuKodea, nace de la necesidad propia debido a la acumulación de trabajo con la prioridad de responder con la mayor brevedad posible. Para así, ser más competente en el mercado. El cual, sólo los más capacitados para obtener en el menor tiempo posible el rendimiento adecuado pueden tener éxito.

Como punto de partida, las siglas vienen del proyecto espacial de la antigua unión soviética que pretendía arrebatarse a la NASA ser los primeros en pisar la luna con programa Apolo, de ahí nace el acrónimo LK-1 el nombre del módulo lunar, pero que nunca llegó a realizar su misión con éxito. Acto seguido, el nombre viene de dos palabras vascas, que significan “obtener código” y de ahí nace el nombre del proyecto y al mismo tiempo su función principal.

Es cierto que, podemos encontrar herramientas similares en el mercado, tanto en programas de escritorio como en navegador, pero todos ellos no tendrán la capa de gestión y curva de aprendizaje tan sencilla, ni la posibilidad de generar nuestro contenido de forma automática o por ejemplo el backup de respaldo de nuestras propias plantillas.

En definitiva, el proyecto consiste en el desarrollo de un portal web enfocado a la generación de plantillas web, tanto para profesionales como para personas que desean tener un portal web propio. El sitio incluye información sobre el proyecto, la propia herramienta está diseñada para satisfacer las posibles necesidades del usuario y mejorar la experiencia del individuo.

1.2. Objetivos del Trabajo

En primer lugar, se establecen los requerimientos de la aplicación, posteriormente se realizan diferentes análisis tecnológicos y competencias. Tanto en, herramientas necesarias para el desarrollo, como tecnologías y los diferentes recursos para poder llevar a buen puerto el desarrollo del mismo, sin olvidarse del diseño de la aplicación. Tanto técnico, funcional y visual. Con el objetivo de implementar un entorno adecuado para poder tener un entorno de trabajo. A fin de, poder competir de tú a tú con diferentes competidos del mercado actual.

La aplicación estará implementada con el CMS Liferay^{[1][2][3]}. Portal que une la capacidad de un programador avanzado con las diferentes capas de personalización pudiendo reducir el tiempo de desarrollo donde se crearán Servicios Web, portlets, etc. necesarios para dotar de las funcionalidades que requiera la aplicación. Teniendo como hilo

conductor una base de datos Oracle, unida al backend por medio de Hibernate y desarrollada una API, para obtener una aplicación robusta y segura.

1.3. Enfoque y método seguido

Como se observa en el punto anterior, existen diferentes conjuntos de objetivos, que se tienen que resolver de forma paralela, pero repartidas en tres fases o entregas. Por no olvidar, que tiene que haber un trabajo previo, tanto de estudio de competencias, como análisis y diseño, propiamente dicho.

✓ FASE I: Análisis y Diseño:

En esta primera fase de encarga del diseño, análisis y planificación se la siguiente fase. Por lo que, previamente se realiza un análisis de todos los lenguajes y temas a tratar en el proyecto. Acto seguido, se realiza una instalación del entorno de desarrollo y todas las herramientas necesarias (Java, Liferay, Eclipse, Gradle, OSGI, ORACLE, etc.).

Al mismo tiempo, se realizan bocetos del análisis de la base de datos y las diferentes pantallas y casos de uso que contara el portal, según se puso constatar en la PEC1-2. Por tanto, durante ese proceso se realizan análisis de competencias para mejorar la competitividad. A su vez, se realizan pruebas en el entorno de desarrollo con las implementaciones de la base de datos, API, y peticiones de servicio, adecuación de la experiencia de usuario y planificación de la siguiente fase.

✓ FASE II: Implementación y Desarrollo:

En esta segunda fase, se trabaja la tarea de desarrollar e implementar todos aquellos procesos que unen la base de datos con la vista de la aplicación a través de un entorno de usuario cuidada y una API, por lo que se necesita:

1. Generar un ServiceBuilder-BBDD-API: a partir de, un esquema relacional se implemente el esqueleto de nuestro proyecto. Puesto que, uniremos la parte de backend con la de servicio y toda ella al resto a través de una API.
2. Desarrollo de servicios y portlets: a partir del ServiceBuilder, vamos a generar nuestros métodos de petición/solicitud de servicios sobre los diferentes recursos necesarios del portal y el resto de herramientas a desarrollar en el portal.

- Otros desarrollos: para complementar la implementación se utilizan diferentes herramientas proporcionadas por Liferay para mejorar el entorno de usuario. Creación de formularios de contacto, página de FAQs, etc.

✓ FASE III: Documentación/Finalización:

En esta fase, vamos a incluir todo el proceso de documentación final. El cual, incluye la redacción de la memoria final, presentación en PowerPoint y video demo.

1.4. Planificación del Trabajo

Finalmente, se puede ver la planificación temporal del TFG. Del mismo modo, se puede ver las diferentes fases.

TFG - LortuKodea

UOC
Jesús Blasco Armifiñana

Inicio del proyecto: mi, 18/9/2019

Semana para mostrar: 1

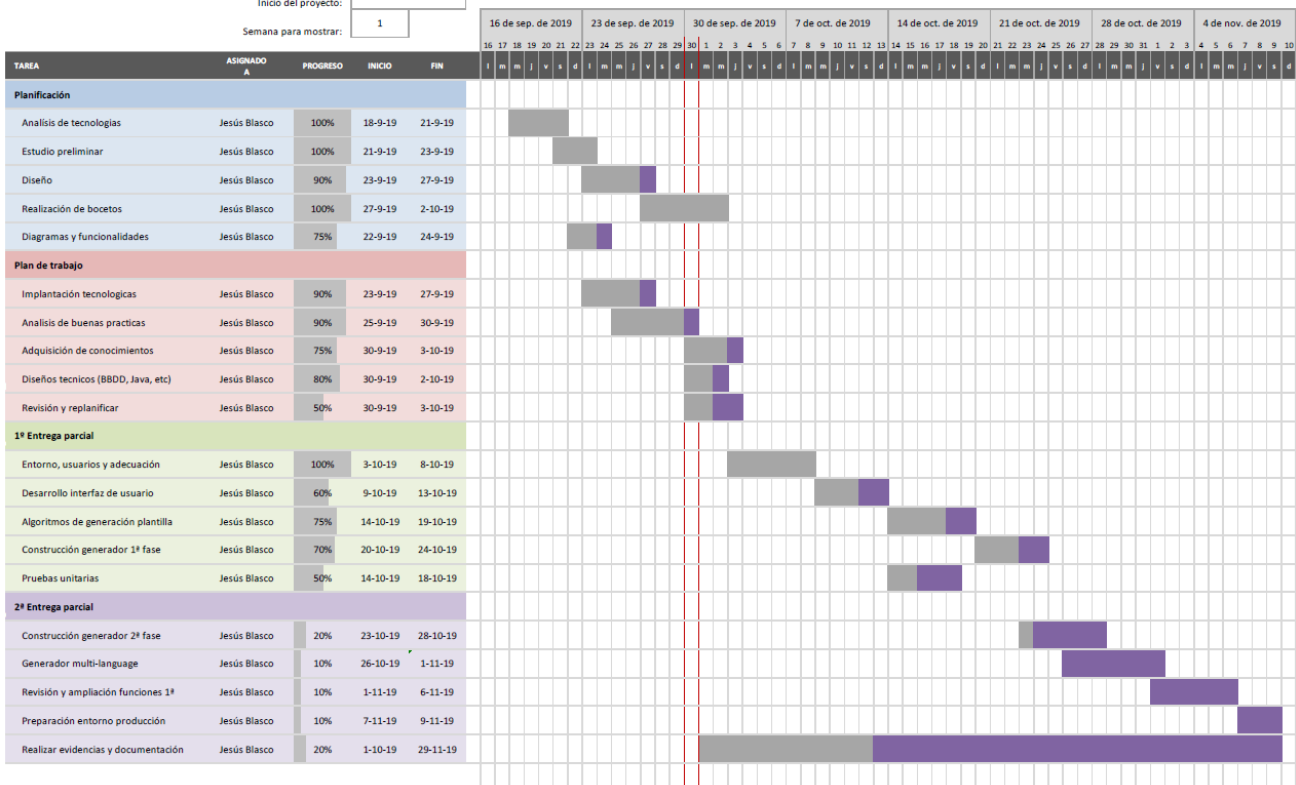


Ilustración 1: Diagrama de Grantt

1.5. Breve resumen de productos obtenidos

En conjunto, el proyecto utiliza y/o genera un conjunto de componentes necesarios para su correcto despliegue y funcionamiento, sobre los prototipos de necesidades que requiere el CMS Liferay:

1. Un servidor de aplicaciones Tomcat en su versión 9.0.17, con la versión de Liferay 7.2.0 CE GA1, que apuntara a una base de datos ORACLE, aunque puede utilizarse MySQL, Como lenguaje principal se utilizara Java 1.8.
2. Un entorno de desarrollo con el SDK propio de Liferay que incluye su IDE, que es una versión de Eclipse con una capa de personalización de los recursos esenciales que necesita Liferay. Por el cual, vamos a poder generar diferentes tipos de módulos (ServiceBuilder, Portlet, API) para Liferay. Del mismo modo, vamos a crear la base de datos al mismo tiempo que el Servicio, ya que la capa de Liferay con Hibernate nos proporciona esa posibilidad autogenerando los procesos.
3. Página de Home: se encontrará elementos estáticos que informaran de las características del proyecto.
4. Página de Proyecto: Información con las características y funcionalidades presentes en la versión actual.
5. Página de Artículos: Información en páginas de terceros sobre el proyecto y noticias propias del mismo.
6. Página de Plantillas (Service Builder, Portlet): Lugar de la implementación más importante el gestor de plantillas que engloba más del 60% de todo el desarrollo.
7. Página de Contacto: Formulario de Contacto, generado con el gestor de formulario de Liferay, con el servidor de correo redireccionado a la bandeja de correo deseada.
8. Página de FAQs: Sitio estático generado con el gestor de estructuras y plantillas de Liferay y que se puede visualizar gracias al gestor/visor de contenidos de Liferay.
9. Página Visor de Plantillas (Portlet): Tabla de administración de las plantillas generadas por parte del usuario en su perfil.

1.6. Breve descripción de los otros capítulos de la memoria

Como se observa en el listado anterior, se implementarán diferentes desarrollos aplicados cada uno a una página diferente. Por lo que, para el corrector desarrollo se realizan diferentes análisis de mercado teniendo como punto de partida las fortalezas de cada uno de los competidores potenciales e intentando dar un mejor punto de vista respecto a los mismos.

Como se menciona, se intenta obtener un modelo que tiene como objetivo servir de código HTML a futuros desarrolladores o usuarios sin conocimiento de programación para obtener estructuras HTML complejas sin necesidad de programar una sola línea de código. A su vez, se utiliza Liferay para dar solución a diferentes problemas de desarrollo que podamos encontrar por el camino, como estabilidad, dinamismo, seguridad, etc.

Destacar, la documentación para una fácil asimilación de los conceptos para su utilización y puesta en marcha, ya que, sirve de guía como a materia ideal para la consulta de dudas que pueda ocasionar, y que se encuentra en los diferentes anexos.

Primero de ello, se realiza un análisis de los procesos que se desean aplicar con las tecnologías aplicadas y el proceso seguido para su correcto desarrollo, despliegue, seguridad, encapsulación e identificación de tipos de usuarios con sus roles y permisos dentro del portal.

Por lo que, se encamina al desarrollo de los elementos principales que interactuaran gracias a la API y al entorno con el servicio de Base de datos aplicado al ServiceBuilder. Para obtener finalmente una aplicación programada para ser lanzada y que a su vez obtiene una planificación de píldoras de actualización con el fin de mantener el proceso de evolución marcado por las necesidades futuras y que no se tendrán en cuenta para esta memoria.

2. Estudio de competencia

Durante un intervalo de tiempo no programado, en virtud de la funcionalidad. Durante el proceso más extenso de espacio para el progreso de la aplicación, se han creado diferentes líneas de análisis de empresas que desarrollan o venden productos de similares características.

Por tanto, el símbolo principal se reubica a mejorar la competitividad en el momento de un hipotético caso de salir al mercado de por sí ya colapsado por muchas herramientas de todo tipo de tecnología y/o plataforma. Con lo que se, realiza un estudio de tecnologías presentes en cada una de ellas y como podrían ser extrapoladas con las tecnologías seleccionadas en el proyecto.

Acto seguido, se realiza una comparativa^[16] de funcionalidades presentes en cada una de ellas seleccionando e ideando el proceso necesario más eficiente de implementar de manera diferente para al mismo tiempo que se igualan funcionalidades buscar un salto de calidad en cuanto a diseño y experiencia de usuario.

Finalmente, se reformula parcialmente las bases fundamentales del proyecto con el fin de poder llegar a mayor número de usuarios potenciales haciendo que elijan nuestra interacción sencilla, pero sin perder robustez o estilo. Por lo que, obtendremos un mejor proyecto final.

En el actual mercado, existe la posibilidad de usar un gestor de contenido (CMS) como WordPress, Joomla o PrestaShop no es necesario. Para proyectos simples y poco complejos que tengan que estar online rápidamente y no se requieran funciones avanzadas. Con lo que, un CMS tradicional puede demorar el lanzamiento del sitio web.

Por tanto, estas plataformas de creación de sitios web proporcionan herramientas que permiten la creación de páginas web sin conocimientos de programación. Del mismo modo, ayudadas de un editor visual (WYSIWYG) para poder añadir contenido y adaptar el diseño. Por norma general, hablamos de sitios como Wix, Jimdo o Weebly, pero también existen programas offline.

Por ese motivo existen diferentes categorías de usuarios más predispuestos a utilizar unas herramientas más offline que online. En definitiva, nos centraremos en un grupo de 10-12 herramientas de ambos tipos para la comparativa. De ese modo nombraremos el programa y a que perfil de usuario está más enfocada.

¹ Bibliografía [16]

Programa (offline/online)	Perfil	LortuKodea
WIX	Creativos	+
Webnode	Multi-lenguaje	+
Jimdo	Múltiple	+
Weebly	Simple	+
Ionos	Variedad de plantillas	+
One.com	Económico	+
WordPress	CMS limitado	-
Shopify	Ecommerce	-
Webs.com	Anticuoado	-
Mobirise	Offline	-
Adobe Muse	Offline	-
Website X5	Windows	-

En esta tabla comparativa, lo que pretendemos es conocer que podemos adquirir y mejorar para nuestro proyecto del resto de competidores, representado con el signo (+) aquello que nos interesa y con el (-) con aquello que nos restaría valor.

2.2. Ventajas y desventajas

De nuevo, vamos a ver una tabla comparativa de aquellas herramientas que nos aportan nuevas características y nos suman de las que nos van a poder restar para nuestro fin más próximo.

Ventajas soluciones ONLINE	Ventajas soluciones OFFLINE
<ul style="list-style-type: none"> ➤ No hace falta que instales nada en tu ordenador. ➤ Puedes trabajar desde cualquier lugar (con internet) y con cualquier dispositivo. ➤ Las actualizaciones son automáticas. ➤ Puedes olvidarte de usar clientes FTP (para subir archivos al servidor), esto no es necesario. ➤ Incluyen el hosting para tu sitio web. ➤ No hay que comprar ningún software. ➤ La mayoría de creadores online ofrecen planes gratuitos con los que probar su herramienta. 	<ul style="list-style-type: none"> ➤ Podrás trabajar en tu sitio web sin tener conexión a internet. ➤ Puedes acceder al servidor donde tu sitio web está alojado. Esto da mayor flexibilidad, pero también puedes desbaratar tu sitio web si no sabes lo que haces. ➤ No dependes tanto del proveedor del creador de sitio web.

Como podemos ver, no hay necesidad imperiosa de encontrar desventajas, ya que cada uno de los métodos es efectivo para cada caso en especial, previo análisis. Con lo que, nosotros vamos a desarrollar como idea principal será una herramienta web, podemos definir qué; estamos creando un CMS web para obtener estructuras completas de sitios web con una herramienta moldeada para fomentar la creatividad gracias a su presentación y diseño. Todo ello, unido a que nosotros mismos estamos utilizando un nivel de CMS por encima de lo querido por parte de nuestros usuarios potenciales para hacerles su trabajo de la mejor y más seguro proceso.

En este caso, podemos ver diferentes webs de referencia con información al respecto sobre la temática más utilizada por parte de los usuarios. Por lo que, únicamente necesitamos sitios web sencillos en principio sin mucha complejidad. Es decir, vamos a pensar en pequeños negocios, portafolios (fotográficos o artísticos) y proyectos web de particulares.

Si queremos ampliar el horizonte, normalmente se integran sistemas de blogs y redes sociales. Por supuesto, el proyecto tiene que ser multi-lenguaje. Por consiguiente, podemos ofrecer servicios de gratuitos y una suscripción Premium o tiempo de prueba del servicio. Teniendo en cuenta que, parte de los pilares será la programación de lenguajes de marcas y JS básicos y avanzados junto con hojas de estilos lo más dinámicas y modulares posibles para una correcta eficiencia así poder maximizar los recursos.

A continuación, vamos a ver una tabla comparativa de los tipos de coste por parte del usuario. Por utilizar cada uno de los servicios en las distintas plataformas:

Programa	Precio
*WIX	<ul style="list-style-type: none"> ➤ Desde 5€/mes ➤ Coste dominio y email a parte
*Webnode	<ul style="list-style-type: none"> ➤ Desde 2'95€/mes
*Jimdo	<ul style="list-style-type: none"> ➤ Desde 6'50€/mes
*Weebly	<ul style="list-style-type: none"> ➤ Desde 7€/mes
*Ionos	<ul style="list-style-type: none"> ➤ Desde 10€/mes
*One.com	<ul style="list-style-type: none"> ➤ Desde 1'99€/mes
*WordPress	<ul style="list-style-type: none"> ➤ Desde 3€/mes
*Shopify	<ul style="list-style-type: none"> ➤ Desde 29\$/mes
*Webs.com	<ul style="list-style-type: none"> ➤ Desde 6\$/mes
Mobirise	<ul style="list-style-type: none"> ➤ Descarga gratuita
Adobe Muse	<ul style="list-style-type: none"> ➤ 200€/año
Website X5	<ul style="list-style-type: none"> ➤ Coste por plantilla 10~20€

En conclusión, tendremos prácticamente desde un inicio descartados proyectos complejos ya que no se dispone de tiempo suficiente para el desarrollo de herramientas funcionales capaces, pero como norma general se utilizarán los patrones más utilizados como hoja de ruta para mejorar las ventajas que hacen destacar a los principales competidores en nuestro beneficio.

2.3. Fortalezas y funcionalidades

Respecto la planificación inicial, tomada en la primera entrega se planifico un periodo de desarrollo hasta el 30 de noviembre aproximadamente, con un periodo de margen de 80 horas (2 semanas), como margen para recalcular y dado el caso evitar retrasos con algún error importante.

Por consiguiente, se consigue realizar las mejoras y cambios con soltura dentro del periodo planificado como “tiempo de reserva” para la mejora de la herramienta en una fase de re-planificación una vez finalizada la primera fase, sino que se consigue eliminar otras tareas con esa optimización realizada durante el análisis de competencia y reevaluación.

Como resultado del análisis de competencias y virtudes del resto de herramientas se han revisado los modelos y diseños previstos en una fase inicial, con lo que se toma la decisión de revisar las tablas del esquema de bases de datos, con el fin de eliminar, optimizar y añadir nuevas tablas y relaciones.

Con estos cambios en la estructura de datos aumentaremos el control de datos, eliminaremos tablas intermedias y optimizaremos la gestión de registros con una nueva tabla que nos proporcione la agrupación de registros de plantillas mejorando las funcionalidades.

A consecuencia de, los cambios en los datos, nos vemos obligados a recompilar todo el servicio añadiendo el proceso de agrupación en las peticiones de base de datos. Por lo que, añadimos las acciones necesarias para insertar los grupos de la mejor manera posible. Además de, por parte del usuario en el apartado de la vista, con lo que renovaremos funcionalidades visuales por parte del usuario.

Por otro lado, se han añadido funcionalidades al primer diseño del generador de menú. Con las dos nuevas funcionalidades de paleta de estilos, la opción de añadir accesos a perfiles de redes sociales más populares. Todo ello, unido a un entorno de usuario cuidada podemos presentar un entorno de usuario muy competitivo.

Finalmente, con el objetivo de dar un aspecto más profesional y cercano con la obtención de feedback se desarrolla un formulario de contacto. Con las herramientas que Liferay proporciona. Además, se genera una plantilla para la creación y administración de la página de preguntas frecuentes. Con estas nuevas funcionalidades ganamos en accesibilidad, conocimientos y cercanía de cara al usuario.

3. Análisis de la aplicación

El proyecto consiste en el desarrollo de un portal web tecnológico ^[17] sobre la construcción de plantillas estáticas, pero responsivas a cualquier dispositivo con el que se visualice.

3.1. Requisitos funcionales

El portal deberá disponer de un menú principal, con los siguientes apartados:

- Información sobre el proyecto.
- Novedades respecto al mismo.
- Artículos de otras herramientas compatibles/complementarias.
- Formulario de contacto con los administradores,
- Acceso al perfil del usuario.
- Herramienta de creación de plantillas.

Por otro lado, no estando contemplado en la planificación inicial como hitos complementarios:

- Vinculación del registro e inicio de sesión con usuarios de google y/u otros.
- Desarrollar el sitio en al menos 2 idiomas, por defecto.

Además, al tratarse de una herramienta de gestión, nos encontraremos con el inconveniente de la gestión de permisos entre la parte pública y privada enfocada a la gestión del portal. Por tanto:

- El lado privado será utilizado por el administrador y usuarios con permisos de edición del contenido, es decir permisos de redactor.
- El lado público, será de uso común del usuario genérico, al que está enfocado el proyecto: tanto profesionales del sector como colectivos que quieran un sitio web sin conocimientos técnicos.

3.2. Roles

Con todo ello, podemos decir que perfiles de usuarios tendrá el portal y cuál será su rol:

- **Administrador:** usuario con permisos de administrador, configuración, gestión del resto de usuarios del sitio.
- **Redactor:** usuario con permisos de creación y edición de contenidos del portal como, edición de noticias, FAQs, consultas.
- **Usuario:** únicamente con permisos de generación de plantillas y visualización del resto de contenidos públicos.

- **Invitado:** usuario que no identifican y que por tanto solo podrán visualizar el contenido público, una vez quieran generar o iniciar el proceso de creación tendrán que registrarse.

3.3. Configurar Permisos

En el apartado de configuración de permisos del portal Liferay vamos a poder encontrar un número importante de configuraciones y roles que podremos configurar según qué acciones queremos que los diferentes usuarios queremos permitir.

En nuestro caso, de momento como vamos a utilizar para los desarrollos propios no tendrá ningún tipo de restricción no es necesario personalizar este aspecto. Únicamente, identificar que en cada portlet en su apartado de configuración podremos cambiar estos criterios si lo deseamos.

3.4. Casos de Uso

Los casos de uso [6] son la descripción de los pasos que queremos que realicen los usuarios, para poder utilizar de la mejor forma nuestra herramienta. En los siguientes apartados se muestran diferentes casos de uso pertenecientes al portal en función de los requisitos que se han definido en los actores y roles que intervienen.

❖ Diagrama: Acceso al sitio web:

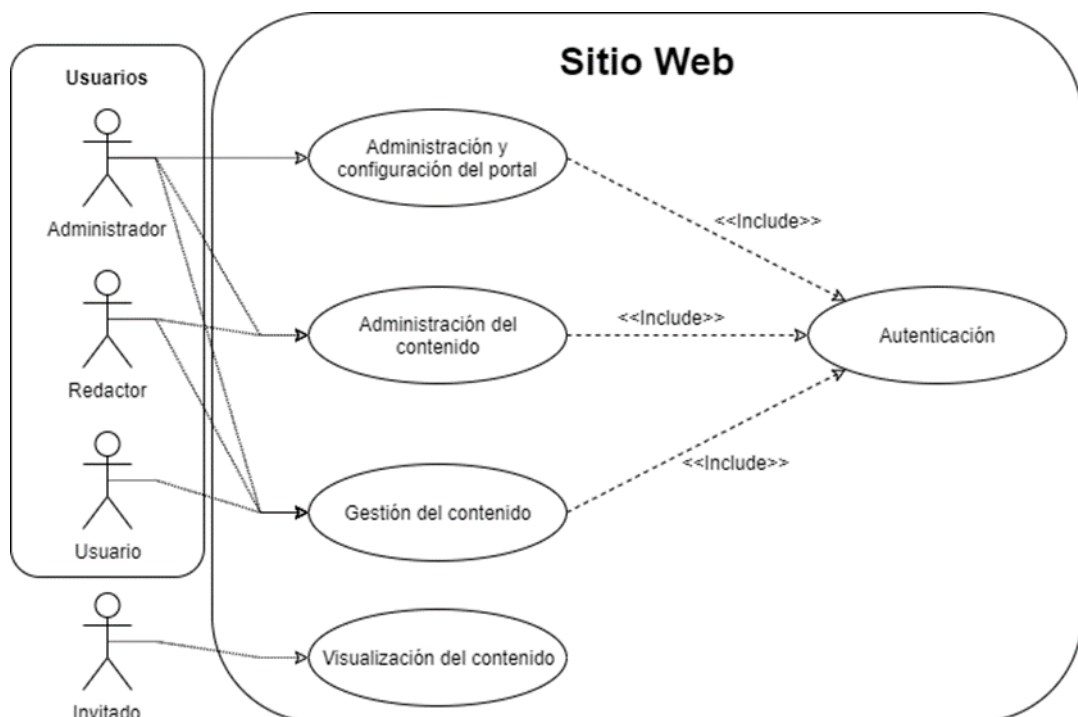


Ilustración 2: Primer caso de uso

- Identificación de los usuarios:

Descripción	Permite a los usuarios autenticarse en el sitio. Una vez identificados, los diferentes tipos de usuarios obtienen acceso sobre los diferentes permisos de acceso en los elementos del portal y en función del perfil/rol
Actores implicados	Administrador, redactor y usuario
Precondición	Estar dado de alta en el sistema de usuarios del portal
Postcondición	Acceder a la parte privada del sitio
Alternativas y excepciones	El usuario no puede identificarse hasta haberse dado de alta

- Administración y configuración:

Descripción	Permite a los usuarios con permisos administrar y configurar por completo el sitio y definir los demás permisos de los usuarios
Actores implicados	Administrador
Precondición	Estar dado de alta en el sistema de usuarios del portal
Postcondición	Gestionar el contenido del portal
Alternativas y excepciones	Los usuarios redactores no podrán administrar el portal para la gestión de otros usuarios, solo podrán gestionar y administrar el contenido de la parte pública y servir de respuesta a las consultas del resto de usuarios

- Gestión del contenido:

Descripción	Permite a los usuarios con permisos gestionar el contenido y servir de primer filtro entre el resto de usuarios con el usuario administrador
Actores implicados	Administrador y Redactor
Precondición	Estar dado de alta en el sistema de usuarios del portal
Postcondición	Administrar el portal
Alternativas y excepciones	Tiene que existir el sitio web

- Visualización y generación:

Descripción	Permite a los usuarios con permisos ver el contenido y sus registros en el sitio
Actores implicados	Administrador, Redactor y usuario
Precondición	Estar dado de alta en el sistema de usuarios del portal
Postcondición	Visualizar el contenido del portal
Alternativas y excepciones	Tiene que estar dado de alta en el sitio para poder visualizar y generar plantillas

- ❖ Diagrama: Gestión de los contenidos del portal:

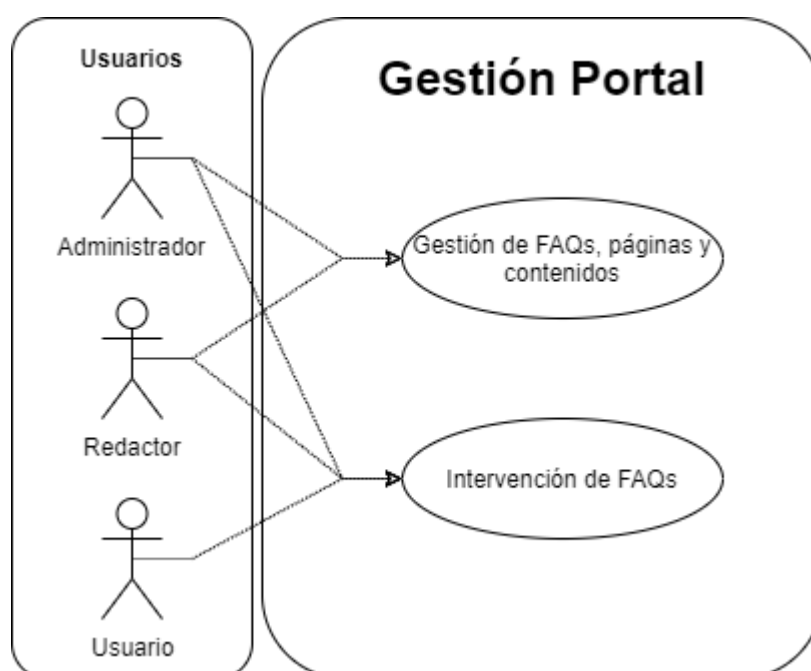


Ilustración 3: Segundo caso de uso

- Gestión de FAQs, páginas y contenidos:

Descripción	Permite a los usuarios con permisos gestionar el apartados de FAQs, páginas y otros apartados de contenidos, como noticias y blogs
Actores implicados	Administrador, Redactor
Precondición	Estar dado de alta en el sistema de usuarios del portal
Postcondición	Gestionar páginas y contenidos del sitio

Alternativas y excepciones	Tiene que estar dado de alta en el sitio para poder gestionar y añadir contenidos
-----------------------------------	---

- Intervención en los apartados de novedades:

Descripción	Permite a los usuarios con permisos intervenir y/o añadir comentarios tanto en noticias, novedades y Blog
Actores implicados	Administrador, Redactor y usuario
Precondición	Estar dado de alta en el sistema de usuarios del portal
Postcondición	Intervenir en los apartados donde se permiten estos comentarios
Alternativas y excepciones	Tiene que estar dado de alta en el sitio para poder añadir comentarios

4. Diseño de la aplicación

Para poder realizar el diseño de la aplicación es necesario introducirse previamente en ciertos conocimientos tanto tecnológicos como de lenguajes de programación que van a utilizarse durante el periodo de desarrollo de todo el proyecto.

Por tanto, vamos a desarrollar la primera decisión que fue adoptar la tecnología o CMS de Liferay como base del proyecto, gracias a las infinitas posibilidades que ofrece y su versatilidad para construir herramientas y portales.

Por lo que, vamos a explicar la estructura que tendrá nuestro portal en Liferay, que herramientas vamos a necesitar para alcanzar el primer hito del proyecto.

4.1. Arquitectura Liferay

Liferay es un portal de gestión de contenidos ^[18] ^[19] que contiene tanto una versión de código abierto como una de pago ambas escritas sobre tecnología Java. Ofreciendo a los desarrolladores una plataforma completa para la creación de aplicaciones web, aplicaciones móviles y servicios web rápidamente, utilizando características y frameworks diseñados para un desarrollo rápido, un buen rendimiento y facilidad de uso.

La plataforma base ya está lista y está construida como un contenedor robusto de aplicaciones que puede crear en mucho menos tiempo del que lo haría desde cero. También incluye un conjunto predeterminado de aplicaciones comunes que puede utilizar de inmediato: gestión de la experiencia web, aplicaciones de colaboración como foros y wikis, documentos y medios de comunicación, blogs y más.

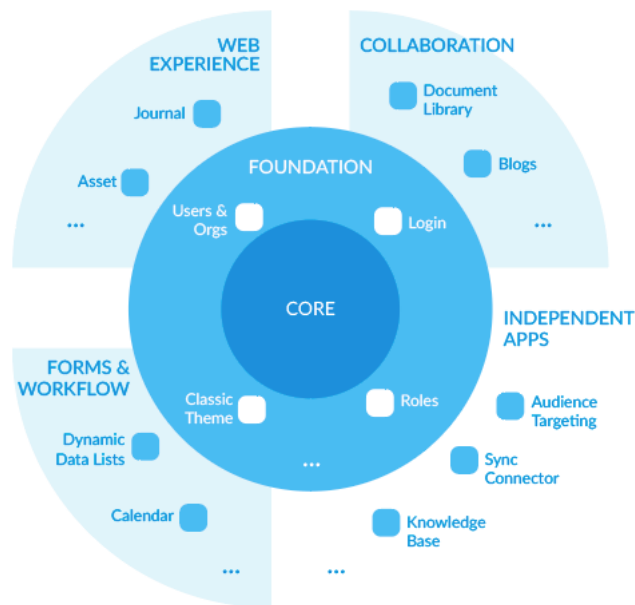


Ilustración 4: Estructura de Liferay

4.2. Tecnología y programas a utilizar

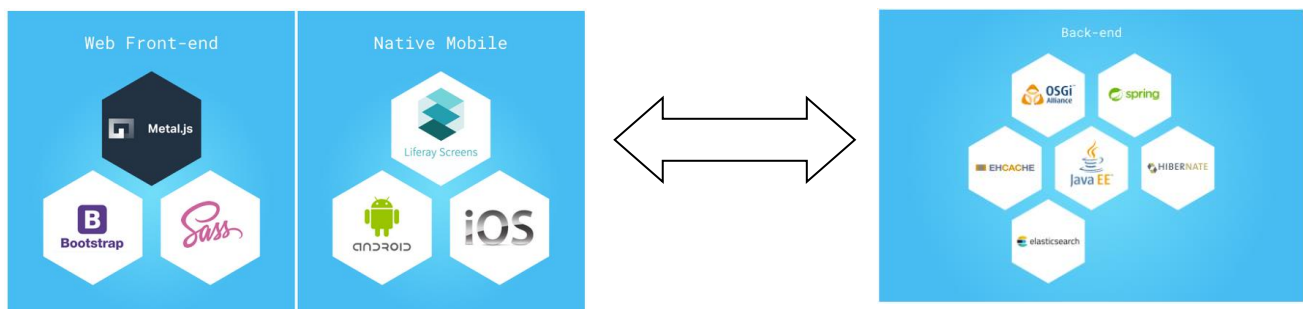
Liferay Portal se basa en la plataforma Java y se puede ampliar añadiendo nuevas aplicaciones, personalizando aplicaciones existentes, modificando su comportamiento o creando nuevos temas. Puede hacerlo con cualquier lenguaje de programación compatible con la JVM, como Java, Scala, jRuby, Jython, Groovy y otros.

Liferay Portal es ligero, puede ser desplegado en una variedad de contenedores de Java EE y servidores de aplicaciones, y soporta una gran variedad de bases de datos. Debido a su capacidad para ser personalizado, puede agregar soporte para más servidores de aplicaciones o bases de datos sin modificar su código fuente: simplemente desarrolle e implemente un módulo con las características que necesita.

Hablando de código y despliegue, aquí tenemos algunas de las formas más comunes de expandir o personalizar las características de Liferay Portal:

- ❖ Desarrollando una nueva aplicación web completa. La forma más común de desarrollar aplicaciones web para Liferay Portal es con portlets, porque se integran bien con otras aplicaciones existentes.
- ❖ Personalizando una aplicación web existente o función. Liferay Portal está diseñado para ser extendido. Muchos puntos de extensión pueden ser aprovechados para modificar el comportamiento existente, y la mayoría de ellos pueden desarrollarse a través de una única clase Java con algunas anotaciones.
- ❖ Creando de un nuevo servicio web para un sistema externo, una aplicación para dispositivos móviles, un dispositivo IoT o cualquier otra cosa.

- ❖ Desarrollando una aplicación móvil que aprovecha Liferay como su back-end, que puede escribir en una fracción del tiempo normal gracias a Liferay Screens y Liferay Mobile SDK.
- ❖ Desarrollando temas para definir la apariencia del portal.
- ❖ Liferay Portal es open source y construido siguiendo un modelo de desarrollo colaborativo.
- ❖ Liferay también está basado en estándares. Esta es una gran noticia para nuestro proyecto, ya que reduce significativamente el bloqueo en Liferay. Eso también nos anima a mejorar constantemente.



La versión actual, Liferay 7.2 de junio de 2019 es la última versión actualizada de la plataforma y como tal contiene infinidad de posibilidades y lenguajes compatibles entre ellos gracias a la capa de integración de Liferay:

- ❖ Desarrollo de nuevas funcionalidades a través de portlets.
- ❖ Incorporación por defecto de más de 70 portlets integrados en el núcleo, que facilitan el desarrollo de portales web.
- ❖ Hibernate para acceder a la base de datos (HSQLDB, MySQL, Oracle, PostgreSQL, etc).
- ❖ Struts, JSF, Facelets, JSTL, etc.
- ❖ Integración con LDAP.
- ❖ Servidor de aplicaciones JBoss, Tomcat, OC4J, Geronimo, Glassfish, Weblogic, Websphere, Jetty.
- ❖ Lucene como motor de indexación y búsqueda de contenidos.
- ❖ Gestión de usuarios, permisos, grupos y roles.
- ❖ Single Sign para la autenticación y autorización una para los diferentes sistemas.
- ❖ Personalización de plantillas, CSS y JavaScript.

- ❖ Basado en los estándares: JSF, JSR, AJAX, Spring, Struts, Tiles, Velocity, WSRP.

4.3. Arquitectura del portal

En primer lugar, a modo de representación visual y siguiendo el patrón marcado desde la web de Liferay se muestra una imagen de representación de toda la arquitectura:



Ilustración 5: Arquitectura de Liferay

4.4. Componentes del portal

Como se indica en la imagen del apartado anterior Liferay lo componen una gran variedad de funcionalidades todas ellas enlazadas mediante portlets preinstalados en el kernel que ayudan a facilitar la construcción del sitio. En este punto nombraremos los tipos de componentes a utilizar y desarrollar en algún caso específico en este proyecto.

- **Portlet:**

Son las aplicaciones de las que está compuesto el portal, encargadas de generar todos los contenidos del mismo. A nivel técnico un portlet está compuesto por al menos una clase Java que implementa el código de las diferentes acciones del mismo. Los portlets están destinados a interactuar con los usuarios gracias a una estructura de peticiones y respuestas.

Por nuestra parte, vamos a desarrollar varios portlets en diferentes puntos del portal para que los usuarios puedan interactuar con la

aplicación y obtener resultados gracias al siguiente componente de Liferay.

- **Service Builder:**

Los elementos service-builder son el elemento raíz del descriptor de despliegue de Liferay para generar los servicios que estarán disponibles desde una aplicación (API). La generación de este código se corresponde tanto a código Java, como Spring o SOAP, siendo la mayoría de clases de persistencia de Hibernate para facilitar el desarrollo de los dichos servicios.

En nuestro proyecto vamos a realizar una base de datos unida gracias a SB con lo que podremos generar una API de forma rápida para poder nutrir a nuestros portlets de contenidos dinámicos y acciones sobre nuestra bbdd.

4.5. Base de datos

En el diseño de la base de datos y gracias a la integración con los servicios e Hibernate, podemos crear todo un sistema de bbdd integrado en el servicio por lo que por medio de xml, se generan las tablas y funciones para la solicitud de servicio sobre esa información. A continuación, detallaremos las diferentes tablas construidas y sus dependencias.

En primer lugar, definiremos que para diferenciar todos los servicios que vayamos a construir nosotros sean identificados por una nueva categoría llamada LK. Al mismo tiempo, Liferay construirá nuestro árbol de clases sobre ese nuevo grupo de servicios.

Para identificar cada tabla crearemos entidades con sus respectivas columnas y características. Acto seguido pasamos a la construcción en primer lugar de cada una de las tablas y una vez las tengamos definiremos su función en el total del sistema.

- **Estructura:**

```
<entity name="Estructura" local-service="true">
  <!-- PK fields -->
  <column name="estructuraId" primary="true" type="Long"></column>
  <!-- Group instance -->
  <column name="groupId" type="Long"></column>
  <!-- Audit fields -->
  <column name="companyId" type="Long"></column>
  <column name="userId" type="Long"></column>
  <column name="userName" type="String"></column>
  <column name="createDate" type="Date"></column>
  <column name="modifiedDate" type="Date"></column>
  <column name="name" type="String" localized="true"></column>
  <column name="description" type="String" localized="true"></column>
  <column name="body" type="String"></column>
  <column name="article1" type="Long"></column>
  <column name="article2" type="Long"></column>
  <column name="article3" type="Long"></column>
```

```

<column name="article4" type="Long"></column>
<column name="article5" type="Long"></column>
<column name="points" type="int"></column>
<finder name="UserId" return-type="Collection">
  <finder-column name="userId"></finder-column>
</finder>
<finder name="GroupId" return-type="Collection">
  <finder-column name="groupId"></finder-column>
</finder>
<finder name="U_G" return-type="Collection">
  <finder-column name="userId"></finder-column>
  <finder-column name="groupId"></finder-column>
</finder>
</entity>

```

En esta tabla quedaran registrados como filas maestras todas las estructuras principales, un total de 12. Las cuales podremos acceder desde la opción Plantillas del menú principal, pudiendo también realizar una valoración. Por todo ello, incluye las diferentes claves y registros genéricos para el resto de tablas que comprenden la bbdd. Por lo que, encontramos: groupId (corresponde al sitio), userId (identificador del usuario), companyId (instancia del sitio), createDate (fecha de creación/actualización).

Por otro lado, las columnas específicas encontramos una descripción que contendrá un resumen del porque la plantilla tiene ese diseño y los fines más comunes se utiliza. Al mismo tiempo, tenemos Body y los diferentes article, que si nos fijamos corresponde al código completo del esqueleto y los articles a las diferentes zonas de escritura dentro de la plantilla.

Finalmente, encontramos el registro de puntuación que realiza un cálculo de la media al momento de realizar la actualización re forma rápida sin coste elevado de memoria. Por último, encontramos los filtros generados para mejorar la visualización en los diferentes desarrollos como el visor de plantillas de cada usuario.

- Registro:

```

<entity name="Registro" local-service="true">
  <!-- PK fields -->
  <column name="registroId" primary="true" type="Long"></column>
  <!-- Group instance -->
  <column name="groupId" type="Long"></column>
  <!-- Audit fields -->
  <column name="companyId" type="Long"></column>
  <column name="userId" type="Long"></column>
  <column name="userName" type="String"></column>
  <column name="createDate" type="Date"></column>
  <column name="modifiedDate" type="Date"></column>
  <column name="estructuraId" type="Long"></column>
  <column name="subRegistros" type="Map"></column>

```

```

<finder name="U_E" return-type="Collection">
  <finder-column name="userId"></finder-column>
  <finder-column name="estructuraId"></finder-column>
</finder>
<finder name="UserId" return-type="Collection">
  <finder-column name="userId"></finder-column>
</finder>
</entity>

```

En el registro, podremos realizar un almacenamiento de todo aquello que el usuario realiza, cuando guarda o descarga el sistema gracias a la API que se a implementado con el ServiceBuilder un registro y a su vez todos los elementos de las otras tablas sabrán donde apuntar sus registros gracias a esté.

Como podemos entender, los subRegistros serán toda la información almacenada en los diferentes departamentos de la plantilla que el usuario podrá escoger si quiere insertar nuevas estructuras o directamente texto enriquecido, insertado en el menú texto.

- Sub-Registro:

```

<entity name="SubRegistro" local-service="true">
  <!-- PK fields -->
  <column name="subRegistroId" primary="true" type="Long"></column>
  <!-- Group instance -->
  <column name="groupId" type="Long"></column>
  <!-- Audit fields -->
  <column name="companyId" type="Long"></column>
  <column name="userId" type="Long"></column>
  <column name="userName" type="String"></column>
  <column name="createDate" type="Date"></column>
  <column name="modifiedDate" type="Date"></column>
  <column name="registroId" type="Long"></column>
  <column name="body" type="String"></column>
  <finder name="UserId" return-type="Collection">
    <finder-column name="userId"></finder-column>
  </finder>
  <finder name="U_R" return-type="Collection">
    <finder-column name="userId"></finder-column>
    <finder-column name="registroId"></finder-column>
  </finder>
  <finder name="SubRegistroId" return-type="Collection">
    <finder-column name="subRegistroId"></finder-column>
  </finder>
</entity>

```

En la presente tabla, podemos encontrar la información que mencionamos en la tabla principal de registros. Teniendo como referencia los subRegistros podremos listar los objetos pertenecientes al registro principal.

- Menú:

```

<entity name="Menu" local-service="true">
  <!-- PK fields -->
  <column name="menuId" primary="true" type="Long"></column>

```

```

<!-- Group instance -->
<column name="groupId" type="Long"></column>
<!-- Audit fields -->
<column name="companyId" type="Long"></column>
<column name="userId" type="Long"></column>
<column name="userName" type="String"></column>
<column name="createDate" type="Date"></column>
<column name="modifiedDate" type="Date"></column>
<column name="registroId" type="Long"></column>
<column name="objetoId" type="Long"></column>
<finder name="UserId" return-type="Collection">
  <finder-column name="userId"></finder-column>
</finder>
<finder name="U_R" return-type="Collection">
  <finder-column name="userId"></finder-column>
  <finder-column name="registroId"></finder-column>
</finder>
</entity>

```

En particular, podemos crear un menú entre 5 estilos diferentes y que podremos personalizar en el nombre de los botones y las redes sociales asociadas.

- Grupo:

```

</entity>
  <entity local-service="true" name="Grupo">
    <!-- PK fields -->
    <column name="grupoId" primary="true" type="Long"></column>
    <!-- Group instance -->
    <column name="groupId" type="Long"></column>
    <!-- Audit fields -->
    <column name="companyId" type="Long"></column>
    <column name="userId" type="Long"></column>
    <column name="userName" type="String"></column>
    <column name="createDate" type="Date"></column>
    <column name="modifiedDate" type="Date"></column>
    <column name="nameGrupo" type="String"></column>
  </entity>

```

Una de las novedades dentro del análisis en la fase de revisión planificada dentro de inicio de la segunda etapa del proyecto, podemos encontrar. Esta tabla se encontrará en la fase de guardado para agrupar nuestras plantillas en grupos para mejorar la paginación y visualización de a que sitio pertenece cada una de ellas, por parte del usuario.

4.5.1.- Reestructuración del Service-Builder

A pesar de, que en el apartado 3.3.7 de la entrega parcia la anterior (PEC2) se realizó un correcto análisis y diseño, en todo proyecto cabe la necesidad de revaluación en la finalización de las diferentes etapas en las que está dividido. Con que, se realiza lo propio con la base de datos quedando redefinida en muy pocos puntos, pero mejorando y optimizando las capacidades.

Por consiguiente, se añade una nueva tabla que mejora la estructuración, organización y funcionalidades. A la vez, que incrementa el modelado de la base de datos y da más coherencia a la información almacenada reduciendo

posible redundancia con tan solo agregar y revisar los objetivos del resto de tablas.

De hecho, gracias a la nueva tabla, junto con su relación en el proceso de registros vamos a poder crear grupos de páginas, Es decir, en el visor de proyectos del panel personal de usuarios con un entorno amigable y sencillo.

A pesar de, tener parcialmente las peticiones de servicio y las tablas montadas y diseñadas se ha conseguido mejorar la funcionalidad de las ya implementadas con lo que las que están pendientes de implementar por completo se han podido eliminar porque ya no son necesarias.

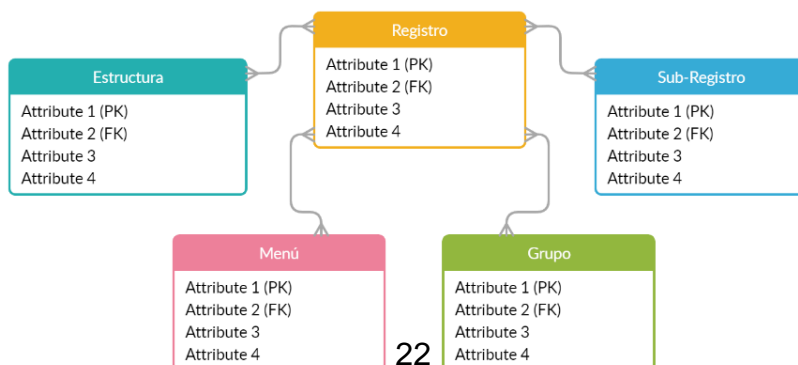
En otras palabras, teníamos en el esquema una definición de 7 entidades, pero hemos añadido una nueva tabla "Grupo". Por consiguiente, una vez refactorizado tenemos:

1º Fase	2ª Fase
Estructura	Estructura
Celda	-
Registro	Registro
Objeto	-
Sub-Registro	Sub-Registro
Menú	Menú
Tipo	-
-	Grupo

No se produce ninguna carga adicional en el proceso de programación. A pesar de, la eliminación de las tablas del servicio no se han perdido líneas de programación. Únicamente, se centralizo el procesado de datos y del modelo.

Con lo que, las tablas pendientes de implementar y que se desarrollarían en la segunda etapa, se han eliminado. Porque, no van a tener significado y función alguna. Entre tanto, se añade una nueva entidad que es sencilla y rápida de adaptar al resto ya desarrolladas.

Por tanto, el diagrama de la base de datos ignorando los campos esenciales e identificando la tabla de registro como la principal podemos ver en un simple diagrama como sería la estructura de nuestra base de datos.



5. Implementación

Como requerimiento previo al desarrollo vamos a tener que necesitar ciertos recursos previos a nuestra disposición, por lo que, tendremos que realizar unas acciones/configuraciones previas:

- Java SDK 1.8
- Oracle Developer
- Liferay Community Edition Portal 7.2.0 GA1
- Apache Tomcat 9
- Liferay Plugins SDK 7.2.0
- Liferay Developer Studio 3.6.1.201-GA2
- Liferay IDE 7.2.0 Mueller build 7200

A modo de aclaración el programa de desarrollo es una versión de eclipse modificada con las propiedades personalizadas y facilidad de gestión de los recursos necesarios para Liferay que ellos mismos distribuyen como Liferay Developer studio de forma totalmente gratuita.

Lo que podríamos hacer sin problemas con nuestro eclipse y descargando desde Marketplace la extensión de Liferay sin problemas. Una vez tenemos todo listo e instalado pasaremos a la preparación de nuestro entorno de desarrollo desde eclipse/Developer studio.

5.1. Instalación entorno de desarrollo

Ahora desde nuestro programa de desarrollo tendremos que crear un servidor Tomcat que apunte a nuestro Liferay.

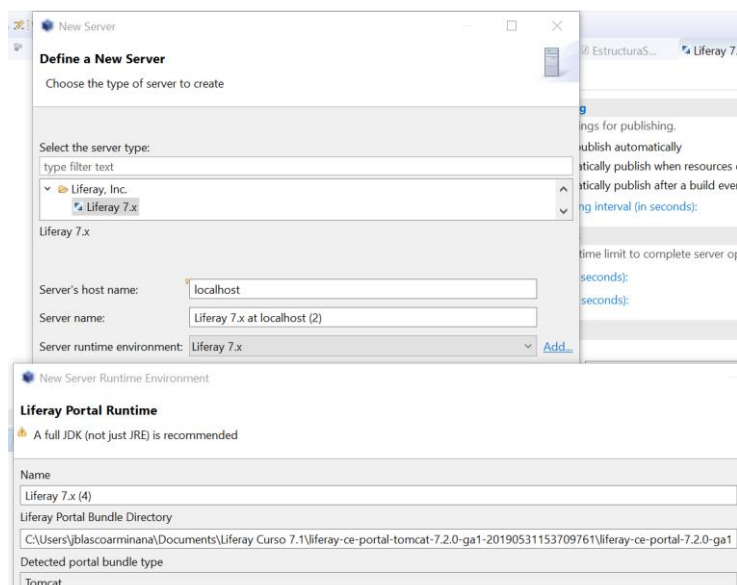


Ilustración 6: Instancia Server

Una vez creado, solo tendremos que añadir nuestros servicios si ya hemos realizado alguno:

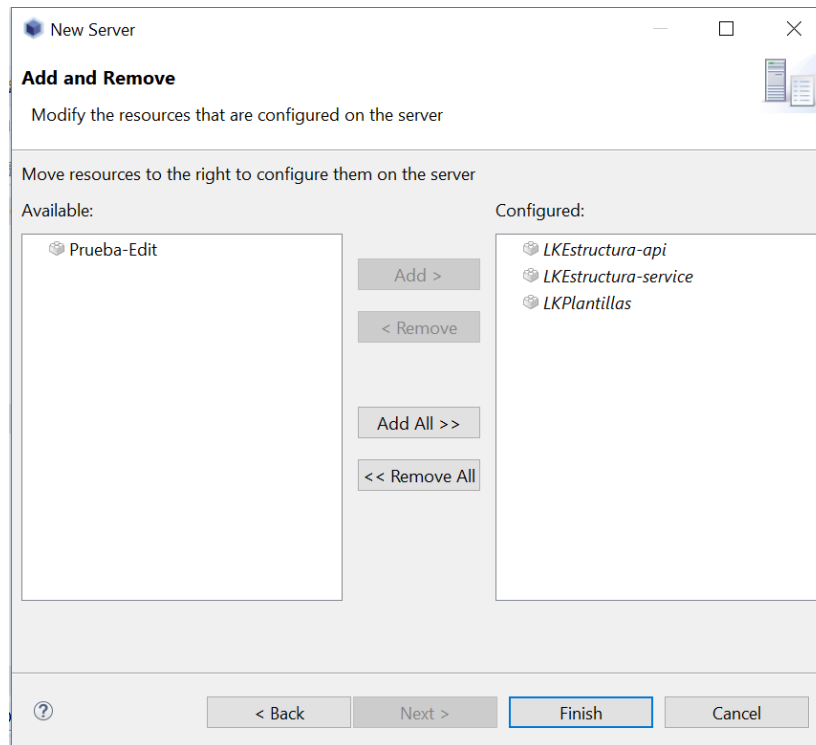


Ilustración 7: Despliegue de Portlets, ServiceBuilders

El proceso de arranque del servidor puede tardar unos minutos y en los primeros pasos nos indicara crear las credenciales del usuario root. Sí todo fue bien ya tendremos nuestro servidor Tomcat con Liferay corriendo, para comprobar que funcione tendremos que ir a localhost o mirar la consola de eclipse y comprobar que en ambos casos se tengan las siguientes imágenes:

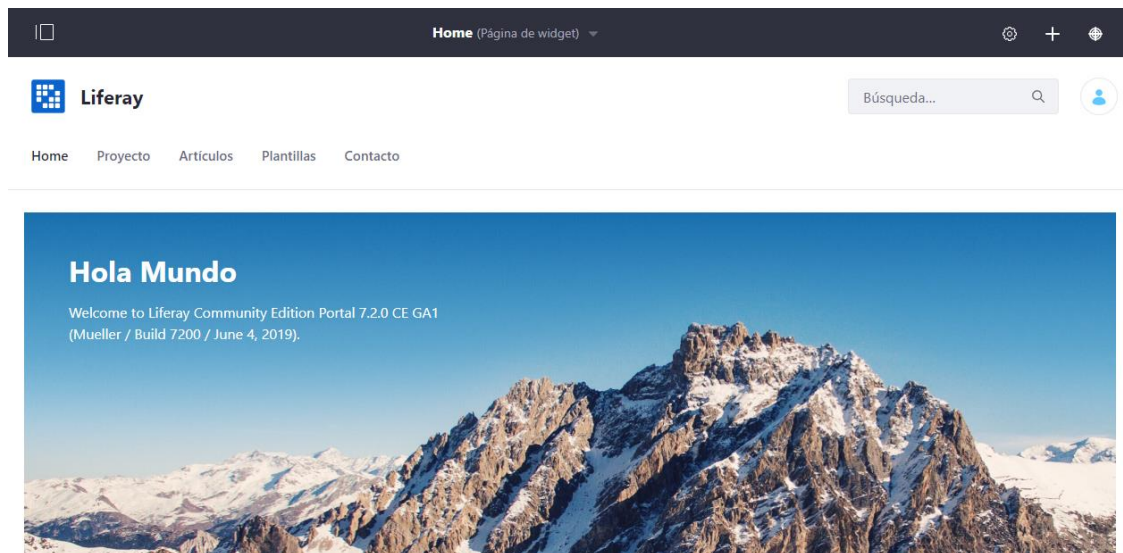
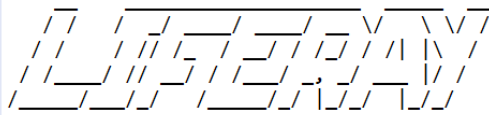


Ilustración 8: Primera página del portal



Starting Liferay Community Edition Portal 7.2.0 CE GA1 (Mueller / Build 7200 / June 4, 2019)

Ilustración 9: Indicativo de que la instancia está iniciada correctamente

Para dar por iniciado el proceso de arranque del servidor y su configuración tenemos las capturas anteriores podemos dar por iniciadas las fases de desarrollo compuesto por dos fases intermedias y la memoria final. Acto seguido, crearemos un workspace para el desarrollo del sitio:

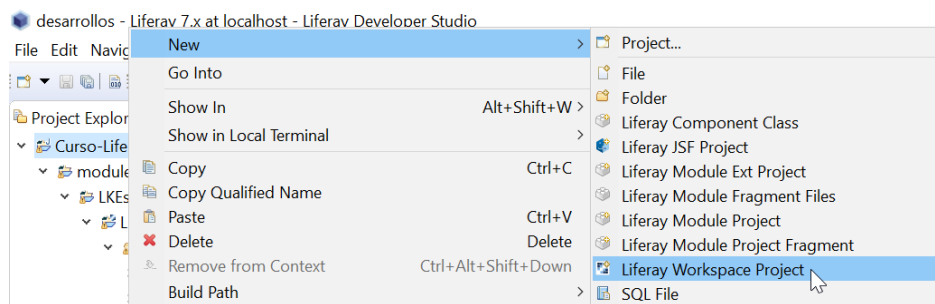


Ilustración 10: Creación de un espacio de trabajo

5.1.1. Creación de las páginas

En este punto vamos a definir la estructura base del sitio. En el proceso de implementación se analizó y definió un patrón base de capas disponibles por defecto en Liferay y que nos brindaba la posibilidad de desarrollar sin complicaciones las herramientas previstas.

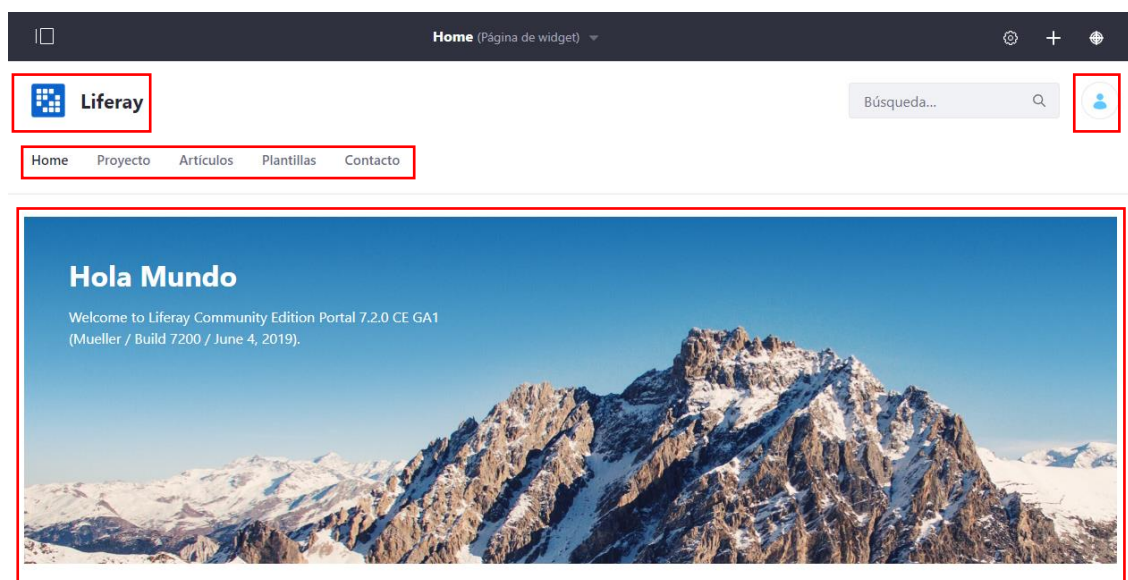


Ilustración 11: Estructuración importante del portal

Podemos ver en un primer encabezado donde se situará el logotipo y nombre del proyecto, al mismo nivel el acceso a las preferencias y configuraciones del usuario una vez logeado. Para posteriormente encontrarnos con el menú que contendrá prácticamente la mayoría de la información y utilidades del proyecto.

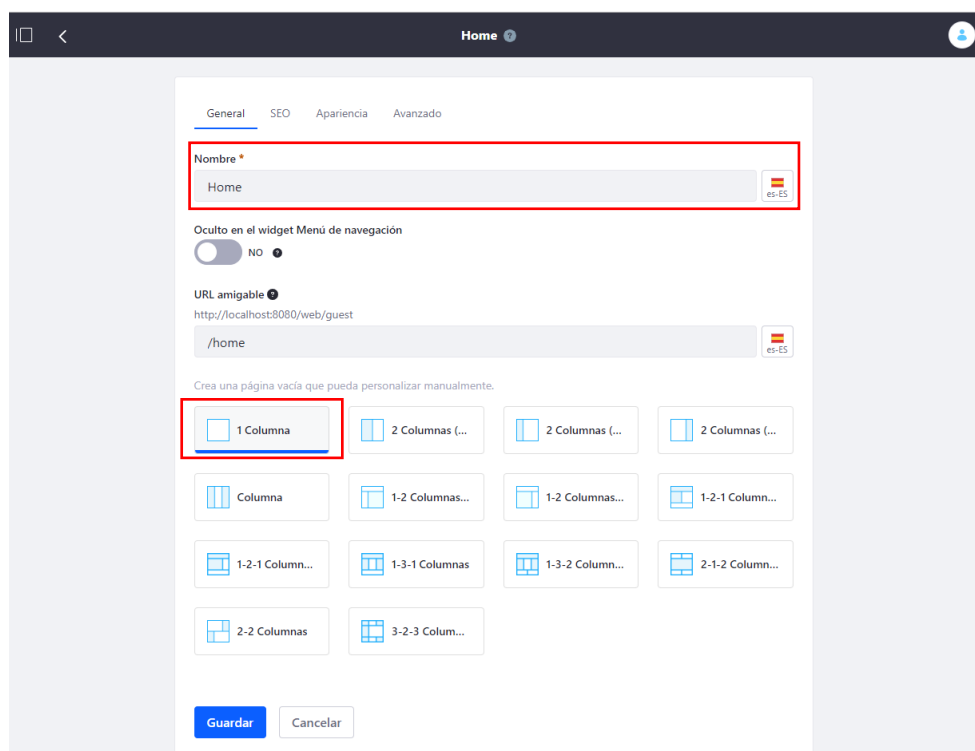


Ilustración 12: Edición de la estructura básica

Por tanto, el resto del contenido estará almacenado en un único contenedor central a una columna que ocupará el mayor espacio posible dependiendo del tipo de resolución del equipo siendo adaptado a cualquier tipo de dispositivo desde del que se visualice.

5.2. Desarrollo del Service Builder

5.2.1. Creación y configuración

Una vez creado el workspace crearemos dentro un módulo de tipo Service-Builder^[5]. Para ello seleccionamos Liferay Module Project y Service-Builder en la select:

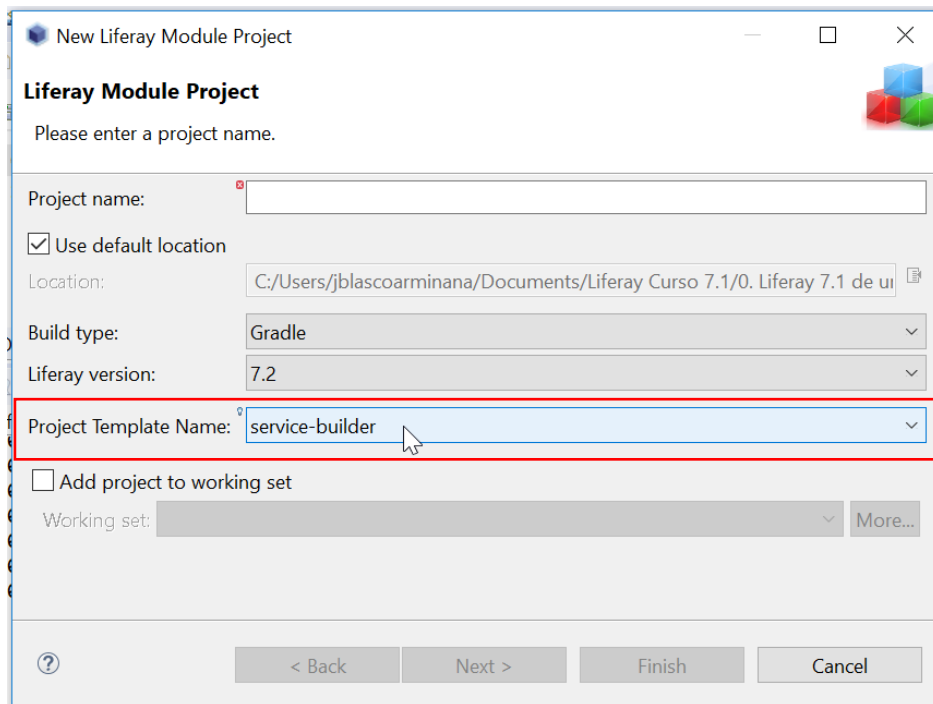


Ilustración 13: Creación del ServiceBuilder

Una vez finalice el proceso de generación del proyecto SB, ya vamos a poder implementar el código xml anterior referente a la base de datos.

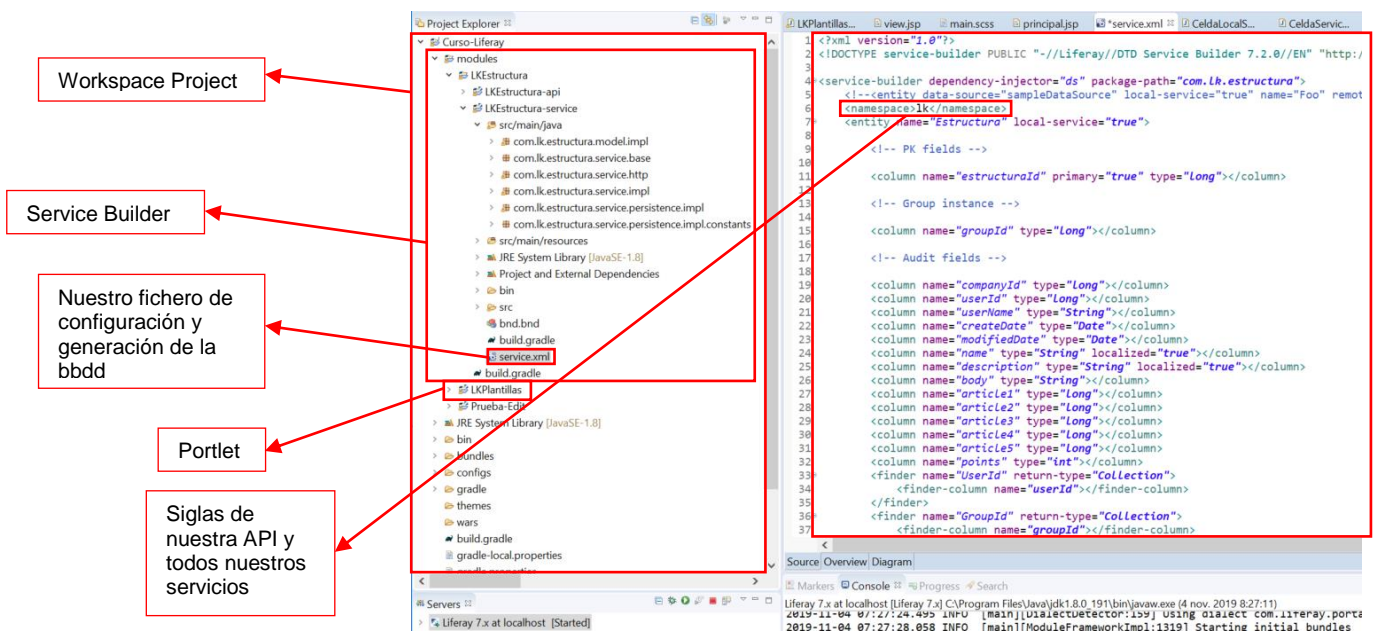


Ilustración 14: Service.xml

Una vez hemos plasmado en el service.xml como será nuestra bbdd, destacar que, si por algún motivo encontramos fallos durante el desarrollo y tenemos que modificar la estructura de las tablas/columnas, etc. solo tenemos que venir aquí cambiar lo necesario y volver a compilar como en la siguiente captura. Importante, desde ahora cada cambio en el Service Builder en la capa de servicio tendremos que recompilar todo para evitar problemas.

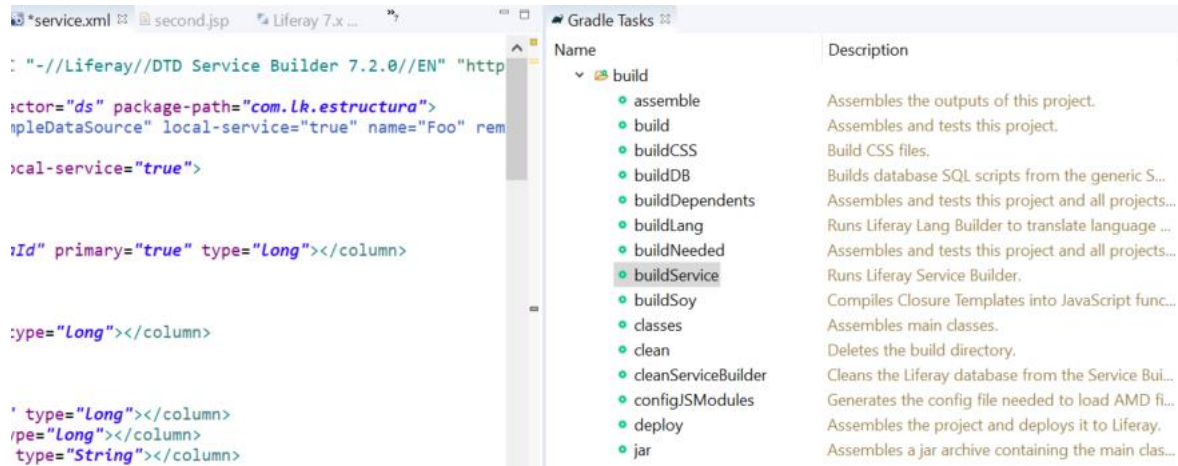


Ilustración 15: Compilador del ServiceBuilder

Gracias a la gran capacidad de unión de un sinfín de tecnologías podemos implementar dependencias rápidamente mucho antes de ponernos a implementar código. Al igual que maven es el Batman de las dependencias y nos facilita mucho la vida y que fue implementada en Liferay en todas las versiones anteriores pero en la 7.2 tomaron la decisión de implementar por defecto Gradle que sería el Ironman también de las dependencias y no sabrías con cual quedarte de ahí la comparación entre ambas casas de superheroes, por mi parte después de mucho tiempo con maven, implemento Gradle al igual que la última versión de Liferay para mejorar e investigar al mismo tiempo.

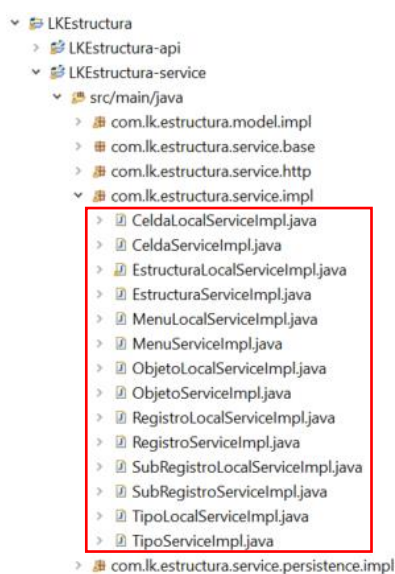


Ilustración 16: Archivos a modificar del ServiceBuilder por parte del usuario

Cuando tengamos nuestro servicio listo para empezar a programar nuestras propias peticiones tendremos que ir a:

Aquí será donde tendremos que realizar todos nuestros métodos y peticiones de servicio para poder ser accesibles desde la API. Perfectamente podemos realizar peticiones de servicio a la parte del servicio, pero no sería un buen desarrollador y si podemos realizar un proceso de buenas prácticas de forma sencilla facilitando la encapsulación y mejorando rendimiento, etc. mucho mejor.

5.2.2. API

Se diseña un servicio en función de las diferentes acciones necesarias para la gestión del contenido y recursos de la base de datos, proporcionando desde la API acceso a todos los recursos necesarios para realizar las

peticiones de servicio requeridas sobre los datos almacenados. Desde liferay nos proporcionan un servicio de pruebas para comprobar el correcto funcionamiento de nuestra API.

Vamos a poder definir en el `localServiceImpl` los métodos necesarios para realizar peticiones, tanto internas como externas, diferentes acciones sobre la base de datos, etc. Pudiendo hacerlas visibles desde la API.

```
57
58= **
59 * addEstructura: creacion estructura principal de plantillas
60 * @param groupId
61 * @param name
62 * @param description
63 * @param serviceContext
64 * @return
65 * @throws PortalException
66 */
67= public Estructura addEstructura(long groupId, Map<Locale,String>name,Map<Locale,String>description,String body,ServiceContext serviceContext) throws PortalException {
68
69     long estructuraId = counterLocalService.increment(Estructura.class.getName());
70     Estructura estructura = super.createEstructura(estructuraId);
71     estructura.setGroupId(groupId);
72     estructura.setNameMap(name);
73     estructura.setDescriptionMap(description);
74     estructura.setBody(body);
75     estructura.setUserId(serviceContext.getUserId());
76     User user = userLocalService.getUser(serviceContext.getUserId());
77     estructura.setUserName(user.getFullName());
78     estructura.setCreateDate(serviceContext.getCreateDate(new Date()));
79     return super.addEstructura(estructura);
80
81
82=
```

Ilustración 17: Ejemplo de contenido en un `LocalServiceImpl.java`

The screenshot shows the JSONWS API testing interface. On the left, there is a sidebar with a search bar and a list of endpoints under the 'Estructura' category, including 'update-estructura', 'set-puntos', and 'delete-estructura'. The main area displays the endpoint `/lk.estructura/add-estructura` with a `POST` HTTP method. Below the endpoint name, the class `com.lk.estructura.service.impl.EstructuraServiceImpl` and the method `addEstructura` are listed. The 'Parámetros' section lists `p_auth` (String), `groupId` (long), `name` (java.util.Map), `description` (java.util.Map), `body` (java.lang.String), and `serviceContext` (com.liferay.portal.kernel.service.ServiceContext). The 'Tipo de retorno' is `com.lk.estructura.model.Estructura`. The 'Excepción' is `com.liferay.portal.kernel.exception.PortalException`. At the bottom, there is an 'Ejecutar' button.

Ilustración 18: Zona de pruebas de las peticiones a la API

Una vez que tenemos el método implementado, realizaremos la inserción en el `servicelimpl` para poder ser visto desde la API y poder ser utilizado de forma

segura con las restricciones necesarias para garantizar la estabilidad y robustez de la bbdd.

```
view.jsp principal.jsp *service.xml EstructuraLocalServiceImpl.java EstructuraServiceImpl.java
49 public class EstructuraServiceImpl extends EstructuraServiceBaseImpl {
50
51 /*
52  * NOTE FOR DEVELOPERS:
53  * Never reference this class directly. Always use <code>com.lk.estructura.service.EstructuraServiceUtil</code> to access the estructura remote service.
54  */
55
56
57 public Estructura addEstructura(long groupId, Map<Locale,String>name,Map<Locale,String>description,String body,ServiceContext serviceContext) throws PortalException {
58
59     return estructuraLocalService.addEstructura(groupId, name, description, body, serviceContext);
60
61 }
```

Ilustración 19: Ejemplo de petición disponible en la API

5.2.3. Despliegue

En el proceso de despliegue del service builder tan solo tendremos que tener en cuenta el proceso mencionado en el punto anterior, botón derecho sobre el servidor.

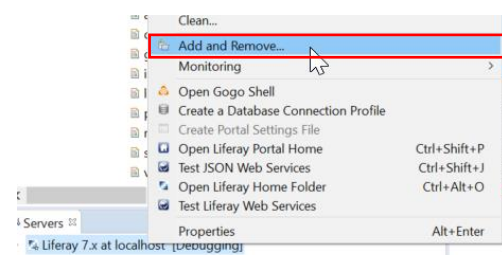


Ilustración 20: Despliegue ServiceBuilder

5.3. Desarrollo de Portlets (modelo)

En la primera fase nos centramos en el proceso de dinamismo de las diferentes plantillas y estructuras que vamos a trabajar en el resto de apartados. El proceso de configuración inicial de un portlet es similar al del Service Builder, salvo que el portlet es una aplicación de cara a realizar acciones el usuario, el cual utilizaremos y/o desarrollaremos diferentes portlets para poder realizar acciones sobre nuestra base de datos con el uso de nuestro servicio.

5.3.1. Creación y configuración

En primer lugar, dentro de nuestro workspace Project crearemos un módulo, proceso idéntico al Service-Builder del punto 4.2.1 salvo que esta vez seleccionamos MVC Portlet.

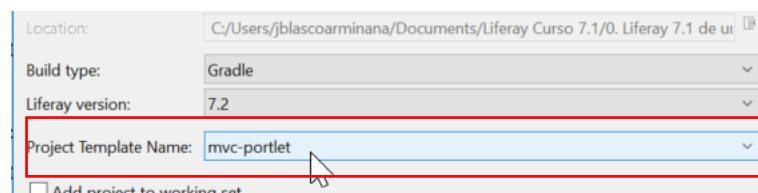


Ilustración 21: Creación portlet

Dentro del proceso de configuración tendremos que comprobar que tenemos las siguientes dependencias tanto en nuestra clase java como en la parte del cliente:

```

1 package com.lk.portlet;
2
3 import com.lk.constants.LKPlantillasPortletKeys;
4
5
6 /**
7  * @author jblascoarminana
8  */
9
10 @Component(
11     immediate = true,
12     property = {
13         "com.liferay.portlet.display-category=category.sample",
14         "com.liferay.portlet.header-portlet-css=/css/main.css",
15         "com.liferay.portlet.instanceable=true",
16         "javax.portlet.display-name=LKPlantillas",
17         "javax.portlet.init-param.template-path=",
18         "javax.portlet.init-param.view-template=/view.jsp",
19         "javax.portlet.name=" + LKPlantillasPortletKeys.LKPLANTILLAS,
20         "javax.portlet.resource-bundle=content.Language",
21         "javax.portlet.security-role-ref=power-user,user"
22     }
23 ),
24 service = Portlet.class
25 )

```

Ilustración 22: Configuración Portlet y referencias necesarias

En el componente de propiedad de Gradle de la clase podremos definir donde se encuentra nuestro CSS, la categoría de nuestro portlet, si puede existir más de uno en el portal, el propio nombre del portlet que se visualizara en el sitio e incluso el nivel de acceso al mismo a nivel de usuarios registrados/identificados.

Por el otro lado, tendremos que realizar la implementación de las siguientes dependencias, recursos necesarios tanto para el correcto funcionamiento de todos nuestros portlets de aquí en adelante como la correcta estandarización de toda la parte web.

```

1 dependencies {
2     compileOnly group: "com.liferay.portal", name: "com.liferay.portal.kernel"
3     compileOnly group: "com.liferay.portal", name: "com.liferay.util.taglib"
4     compileOnly group: "javax.portlet", name: "portlet-api"
5     compileOnly group: "javax.servlet", name: "javax.servlet-api"
6     compileOnly group: "jstl", name: "jstl"
7     compileOnly group: "org.osgi", name: "org.osgi.service.component.annotations"
8     compileOnly group: "com.liferay", name: "com.liferay.osgi.util", version: "3.0.0"
9     compileOnly group: "javax.servlet", name: "servlet-api", version: "2.5"
10    //compileOnly group: "com.google.cloud", name:"google-cloud-translate", version: "1.93.0"
11    compile project(":modules:LKEstructura:LKEstructura-api")
12}
13 repositories {
14     maven {
15         url "https://repository-cdn.liferay.com/nexus/content/groups/public"
16     }
17}

```

Ilustración 23: Dependencias del Portlet

Una vez cargadas las dependencias externas, vamos a cargar las necesarias para aparte de tener un estándar para todo nuestro backend necesitamos que todo funcione correctamente y conectado con la parte visual del proyecto.

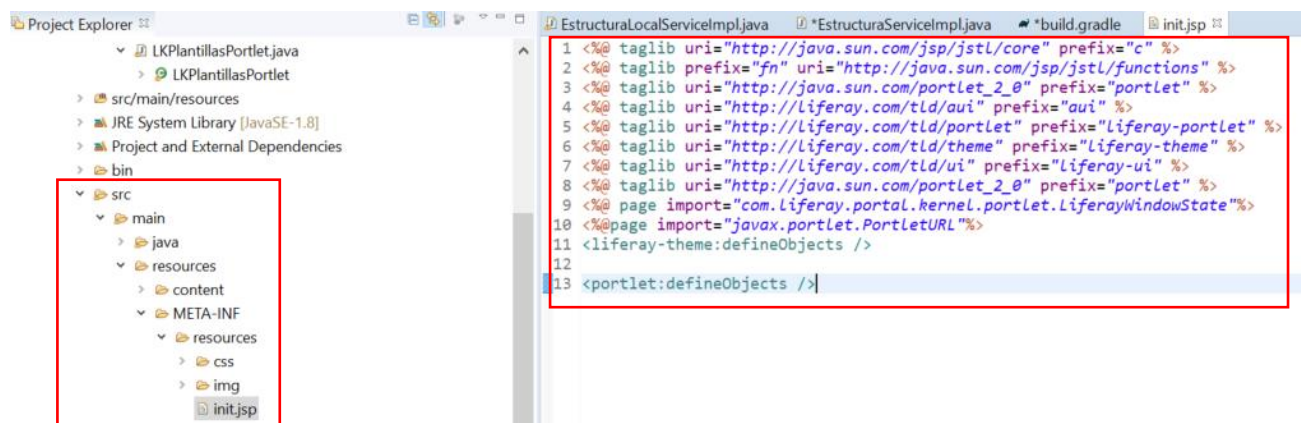


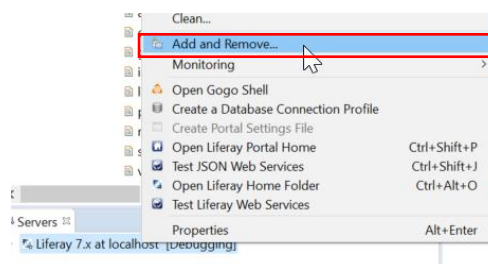
Ilustración 24: Dependencias de la parte Front

5.3.2. Estructura e implantación

Todo listo para empezar, para esta primera parte del portlet de plantillas nos centraremos en la generación y visualización dinámica de las estructuras principales y secundarias de las plantillas y de descargarla, no nos preocuparemos por guardar registro alguno de ellas.

5.3.3. Despliegue

En el proceso de despliegue del Portlet tan solo tendremos que tener en cuenta el proceso mencionado en el punto anterior, botón derecho sobre el servidor.



5.4. Formulario de contacto

En el siguiente apartado, tenemos como objetivo principal la mayor eficiencia respecto las herramientas integradas en el propio Liferay. Disponemos de la generación de formularios completamente integrado en el panel de administración. Vamos a crear nuestro formulario de contacto y configurar el correo electrónico el cual recibirá dichos formularios de contacto.

Respecto el punto de, gestión de correo en el panel de administración del servidor se configurará el correo electrónico que recibirá todas las notificaciones de correo configuradas en el portal. Como requisito indispensable es que el correo pertenezca a un usuario registrado con permisos de administración.

Por consiguiente, se tiene que tener en cuenta los puertos y el gestor correspondiente al servidor de correo en éste caso tendremos que seleccionar el pop y el stmp de google para el correcto envío de los formularios.

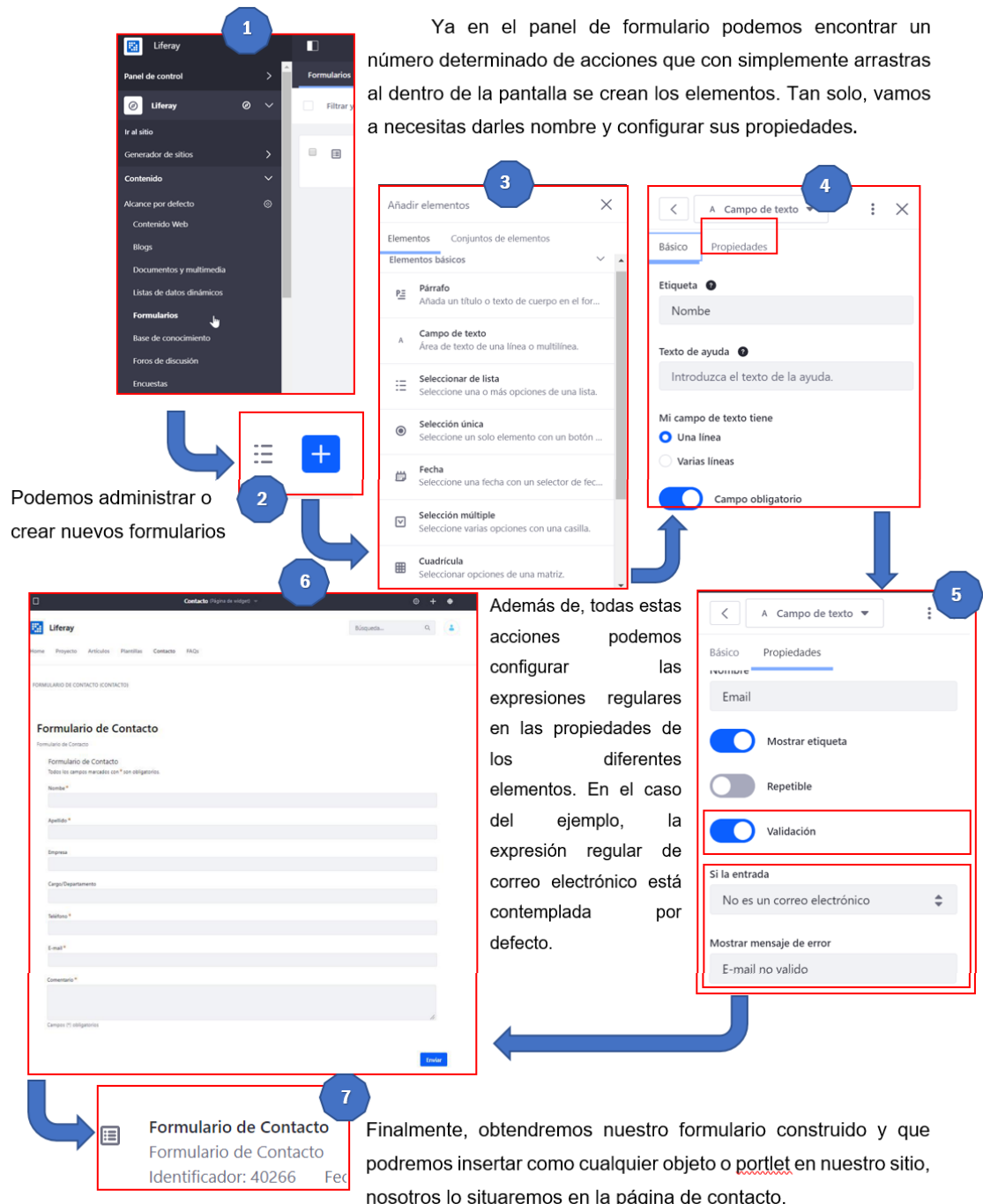


Ilustración 25: Proceso de creación Formulario de Contacto

Una vez, creado el formulario de contacto tenemos que ir al panel de administración del servidor, más concretamente al apartado de Correo. Donde configuraremos el sistema de correo.

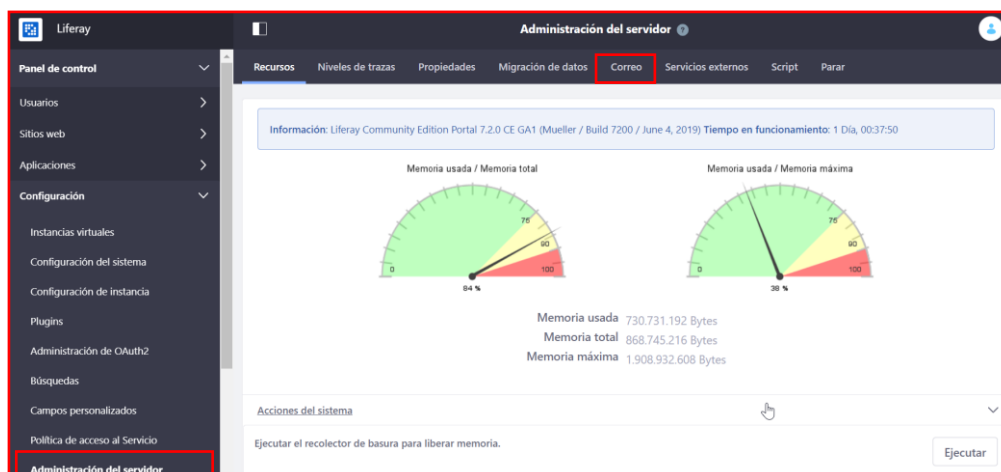


Ilustración 26: Configuración Servidor de correo

Por lo que, en el panel de administración de correo, configuraremos el sistema con los datos correspondientes a un usuario registrado en el sistema con permisos de administración el cual debe tener correctamente su correo electrónico. El sistema de correo se tendrá que configurar de una forma u otra.

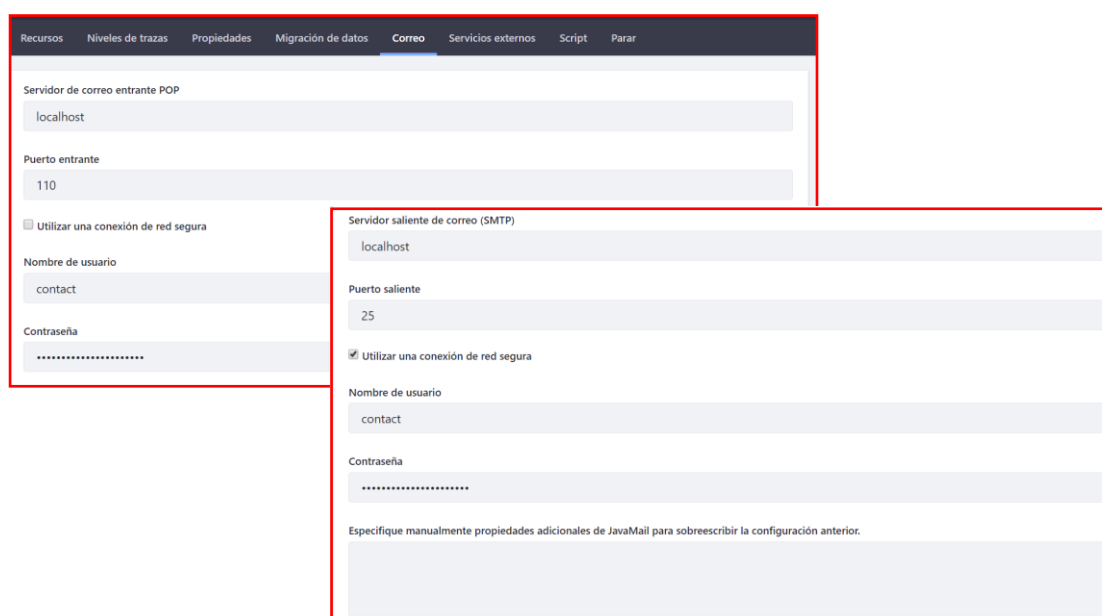


Ilustración 27: Opciones del servidor de correo

5.5. FAQs

En primer lugar, para la creación de nuestra herramienta de FAQs^[4], vamos a necesitar crear una plantilla, un proceso similar al punto de los formularios, salvo que este se llama plantilla en vez de formulario.

En primer lugar, tendremos que crear la estructura web básica para nuestro sistema de FAQs, el cual estará en el panel de control, contenido, contenido web. Para posteriormente seleccionar estructura.

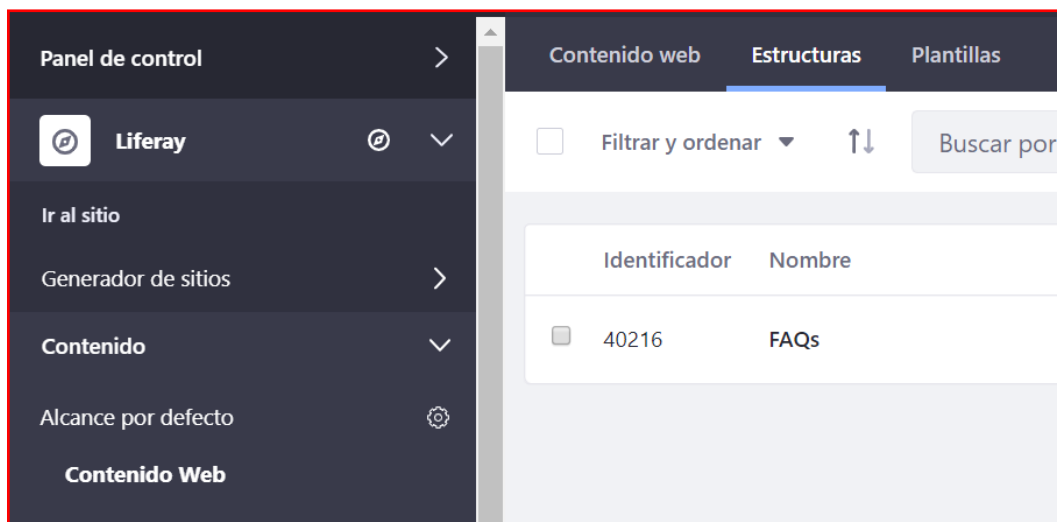


Ilustración 28: Creación plantilla FAQs

Una vez, en el apartado estructuras creamos una nueva y añadimos dos cajas de texto una dentro de la otra con el editor de elementos de Liferay.

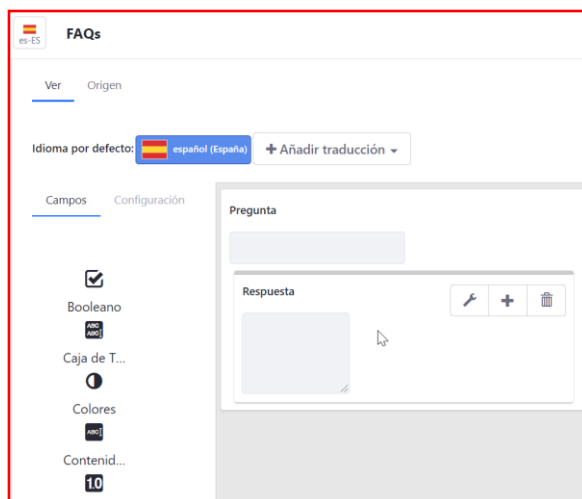


Ilustración 29: Estructura de FAQs

Acto seguido guardamos la estructura y vamos a la pestaña de plantillas y allí, gracias a Fremaker vamos a crear el patrón de la estructura para crear una plantilla que se repita n veces en una misma página de forma dinámica. También, implementaremos el CSS y JS para mejorar la experiencia de visualización.

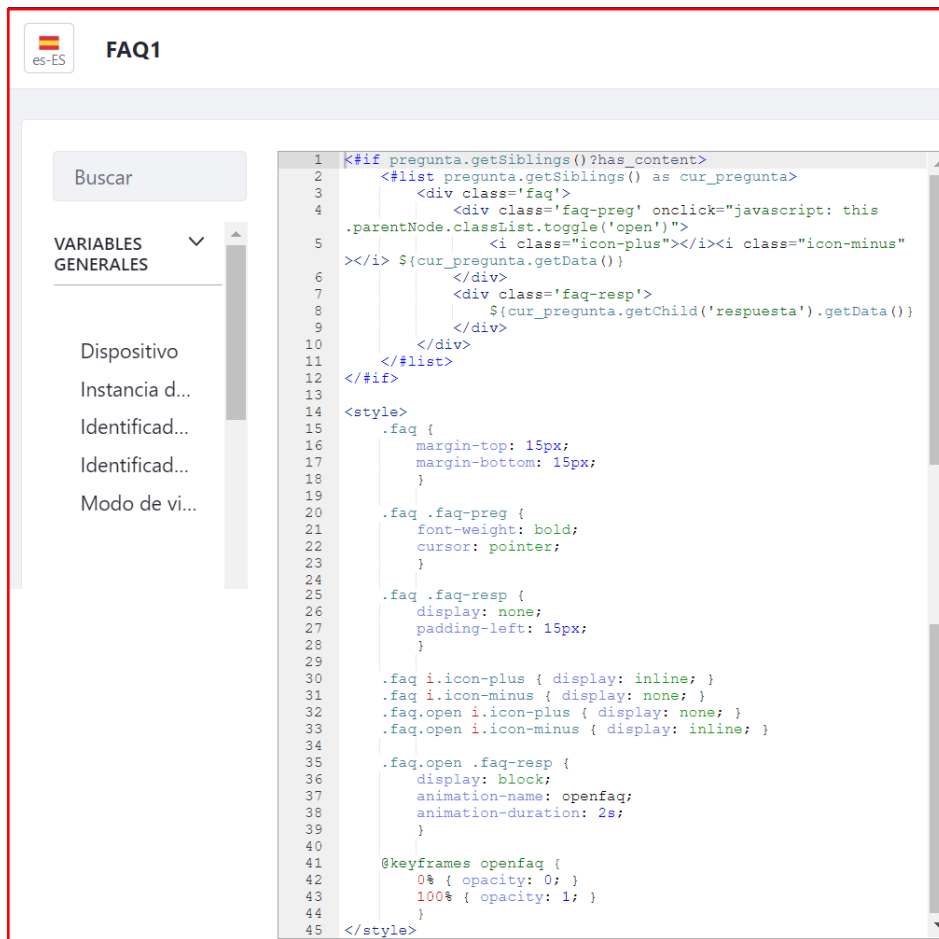


Ilustración 30: Código fuente del estilo de FAQs

Finalmente, guardaremos la plantilla y nos iremos al visor de contenidos en nuestra página pública de FAQs. Es aquí, donde vamos a terminar de construir el contenido definido que contendrá la estructura de la plantilla.

Por tanto, tendremos que desplegar el portlet/aplicación de visión de contenidos en nuestra página de FAQs. Para acto seguido en la pestaña superior izquierda de la misma seleccionar contenido web básico.

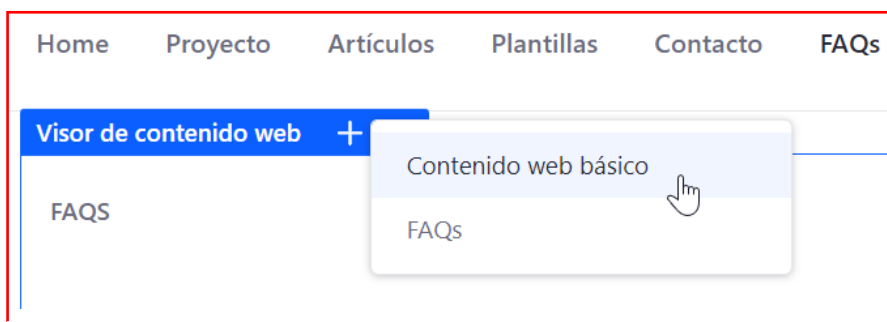


Ilustración 31: Como añadir FAQs al portal

Seguidamente nos indicara si queremos hacer un sitio de contenido estático simple o si queremos añadir contenido a una plantilla, seleccionamos nuestra plantilla y automáticamente nos mostrara nuestra estructura con la posibilidad de añadir tanto como queramos.

Ilustración 32: Multiplicar plantilla automáticamente y añadir la pregunta y la respuesta

Finalmente, podremos ver el resultado después de guardar los cambios, obteniendo un sistema completamente estable dinámico responsivo y fácil de usar por parte del usuario.

Ilustración 33: Resultado FAQs

6. Desarrollo

6.1. Portlet: Plantillas

6.1.1. Seleccionar y diseñar estructura

Por lo que por medio de nuestra API y del método `doView` propio de Liferay vamos a hacer que siempre cargue en la página principal de nuestro portlet una tabla con las estructuras principales de plantillas:

```
@Override
public void doView(RenderRequest renderRequest, RenderResponse renderResponse)
    throws IOException, PortletException {

    List Estructura = EstructuralLocalServiceUtil.getEstructuras(QueryUtil.ALL_POS, QueryUtil.ALL_POS);
    renderRequest.setAttribute("Estructuras", Estructura);

    super.doView(renderRequest, renderResponse);
}
```

Ilustración 34: Código de la petición al cargar la página principal del portlet Plantillas

Con este método cargaremos de forma dinámica las filas de nuestra tabla en forma de objetos de propiedades y que gracias de JSTL vamos a construir de forma dinámica en la vista de nuestra web.

```

<c:forEach var='e' items='${Estructuras}' varStatus="counter" begin="0" end="11">
  <portlet:renderURL var='principal'>
    <portlet:param name='jspPage' value='/principal.jsp' />
    <liferay-portlet:param name="estructuraID" value="${e.estructuraId}" />
  </portlet:renderURL>

  <portlet:renderURL var='puntuar'>
    <portlet:param name='jspPage' value='/view.jsp' />
    <liferay-portlet:param name="estructuraID" value="${e.estructuraId}" />
  </portlet:renderURL>

  <div class="col well" value="" name="" onclick="Location.href='<%=principal%>';">
    <h5>${e.name}</h5>
    
    <CENTER>
      <div id="rate${e.estructuraId}" name="rate${e.estructuraId}"></div>
      <script type="text/javascript">
        $(function () {
          $("#rate${e.estructuraId}").rateYo({
            rating: ${e.points},
            precision: 0,
            spacing: "5px",
            starWidth: "13px"
          });
        });
      </script>
    </CENTER>
  </div>
</c:forEach>

```

Ilustración 35: Código página principal del portlet de Plantillas

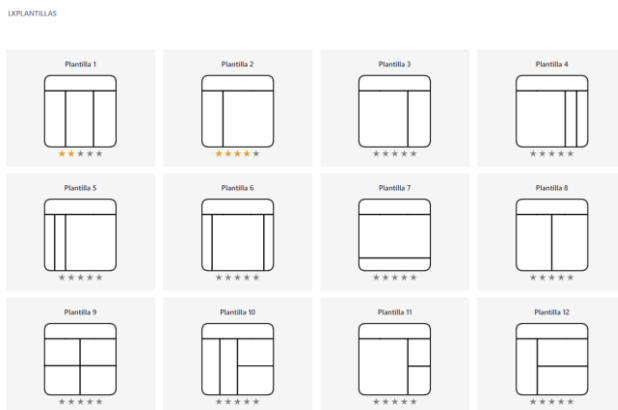


Ilustración 36: Resultado página principal

Con este código vamos a crear un mallado de elementos completamente dinámico y adaptable a diferentes dispositivos. Podemos ver en las imágenes la visualización desde un ordenador y el resultado desde cualquier dispositivo móvil.

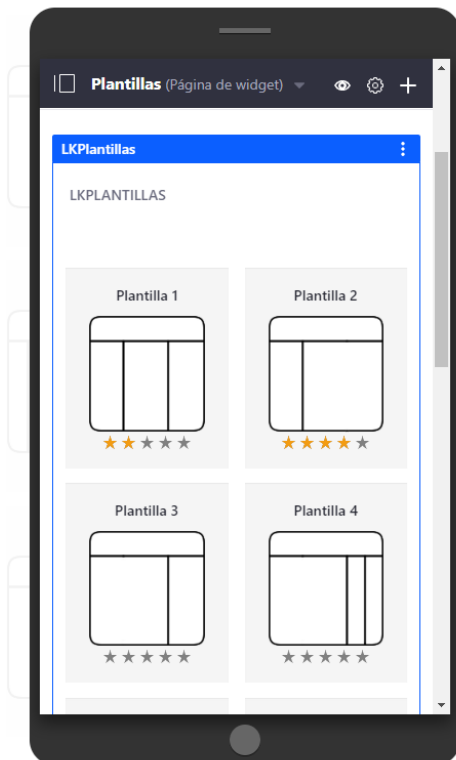


Ilustración 37: Dispositivos móviles

Al mismo tiempo podríamos realizar una valoración de las mismas plantillas, dentro del servicio y del portlet se realizan una acción y un proceso de petición para realizar el proceso de puntuación y valoración. El proceso es sencillo cada

vez que se realiza calcula la media del proceso sin necesidad de disponer de 3 propiedades/valores.

```
public Estructura setPuntos(long estructuraId,int puntos) throws PortalException{
    Estructura estructura = getEstructura(estructuraId);
    int puntosActuales = estructura.getPoints();
    int nuevosPuntos = 0;
    nuevosPuntos = puntosActuales+puntos;
    if(puntosActuales!=0) {
        nuevosPuntos = nuevosPuntos%2;
    }

    estructura.setPoints(nuevosPuntos);
    return super.updateEstructura(estructura);
}
```

Ilustración 38: Llamada en el proceso de votación/puntuación de plantillas

Adicionalmente, podremos seleccionar una estructura principal, para procesar sus subestructuras, estas subestructuras como la principal se cargan nuevamente de forma dinámica y similar a la página principal.

Nuevamente, de forma dinámica y gracias a JSTL vamos a construir nuestras funcionalidades y elementos como propiedades de la página.

```
@Override
public void render(RenderRequest renderRequest, RenderResponse renderResponse) throws IOException, PortletException {
    String jspPage=ParamUtil.getString(renderRequest, "jspPage", "");
    if(jspPage.equals("/principal.jsp)){
        String idestr = ParamUtil.getString (renderRequest, "estructuraID");
        long estructuraId = new Long(Long.parseLong(idestr));
        Estructura estr = null;
        try {
            estr = EstructuraLocalServiceUtil.getEstructura(estructuraId);
        } catch (PortalException e) {
            e.printStackTrace();
        }

        List Celda = CeldaLocalServiceUtil.getCeldas(QueryUtil.ALL_POS, QueryUtil.ALL_POS);

        renderRequest.setAttribute("Celdas", Celda);
        renderRequest.setAttribute("estructura", estr);
    }

    super.render(renderRequest, renderResponse);
}
```

Ilustración 39: Llamada al hacer click sobre una plantilla principal


```

7=<div class="row" id="descripcion">
8     <h3>Selección de estructura secundaria:</h3>
9 </div>
10
11=<div class="row fila-estructura-select-subestructura" >
12  ${estructura.body}
13
14=<c:forEach var='c' items='${Celdas}' varStatus="counter" begin="0" end="15">
15
16  <!-- construccion sub-estructuras /img/estr${counter.index}.png -->
17=<li class="col thumbnail s encapsular" value="${c.celdaId}" >
18
19  <!--
20     
21     <input id="hidden${c.celdaId}" type="hidden" value="<c:out value='${c.body}' />" />
22
23  </li>
24 </c:forEach>
25 </div>
26=<div class="row fila-btn-selectSubEstructura">
27  <div class="col-md-12 text-center">
28    <div class="btn-group btn-group-Lg">
29      <button type="button" class="btn btn-primary">Añadir Texto</button>
30      <button type="button" class="btn btn-primary">Añadir Menú</button>
31      <button type="button" class="btn btn-primary">Guardar</button>
32      <button type="button" class="btn btn-primary">Descargar</button>
33    </div>
34  </div>
35 </div>

```

Ilustración 40: Código página de edición de plantillas

Tan solo nos faltara visualizar que todo esté correctamente construido y montado, por lo que comprobaremos los mismos pasos que el punto anterior.

Selección de estructura secundaria, estas en la caja: uno

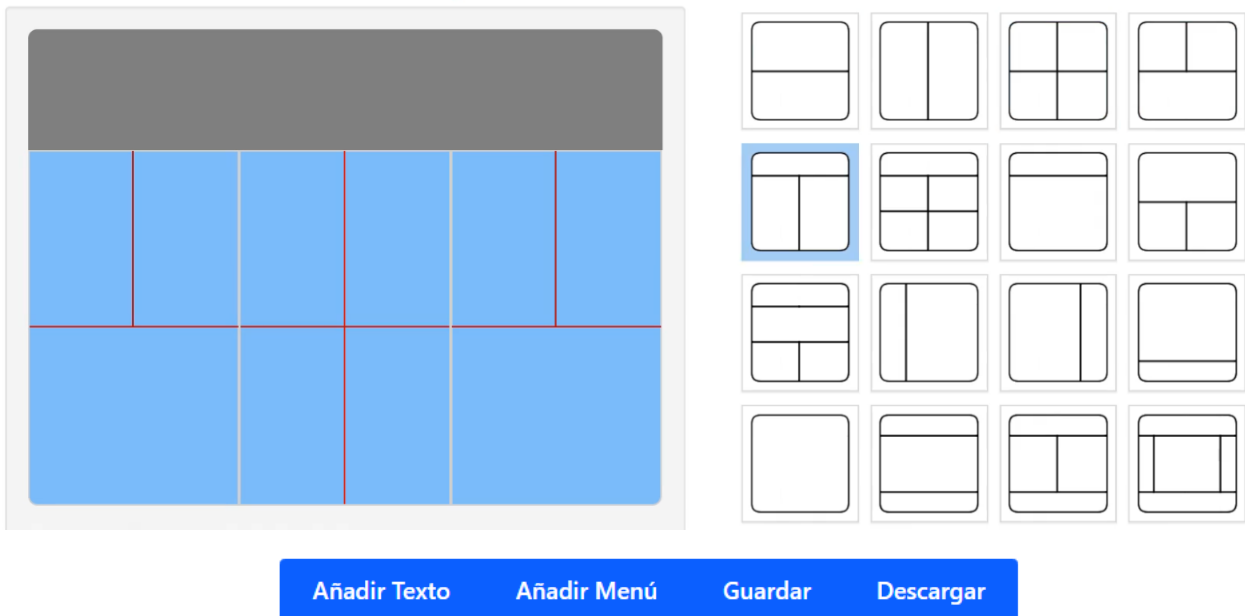


Ilustración 41: Resultado del código anterior

Finalmente, podremos descargar nuestra plantilla, hasta el momento no podemos realizar ninguna otra acción ya que es el límite marcado de la primera fase, pero se han desarrollado otras funcionalidades previas para avanzar y agilizar en las futuras decisiones.

hoy el head plantilla 1



Ilustración 42: Ejemplo de resultado obtenido

El proceso de descargar se compila y genera en formato .zip, teniendo un tiempo de respuesta mucho más rápido que cualquier otro tipo de proceso estudiado para el caso de uso de descarga ya que se realiza una acción de procesado de datos en el buffer del proceso y que en apenas unas milésimas de segundo genera y compila el código para empaquetar un zip e iniciar el proceso de descarga.

6.1.2. Añadir/Editar texto

Junto al punto anterior, pero sobretodo el más importante nos encontramos con el punto de inflexión de toda la herramienta donde el usuario podrá editar texto, añadir imágenes y video en la nube ^{[7][8]} o por medio de enlaces todo el contenido de su sitio web.

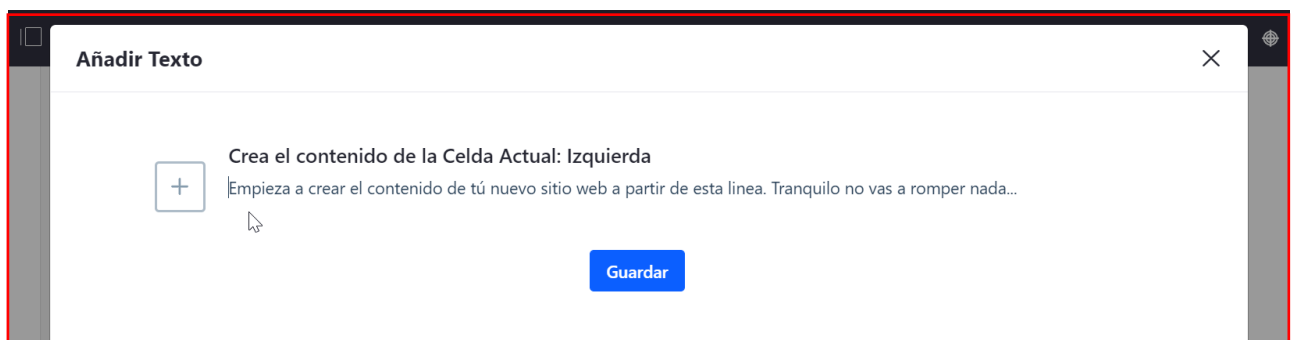
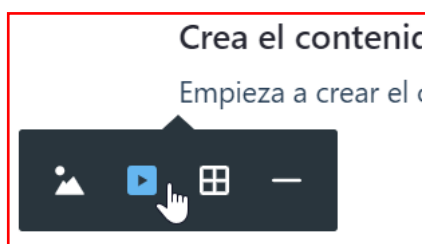


Ilustración 43: Insertar texto enriquecido



Al situar el cursor como un editor de texto el usuario observara aparecer el símbolo de "+", que al presionar mostrara las acciones multimedia y de estructuración por medio de tablas y líneas de separación

Ilustración 44: Insertar contenido multimedia en la nube

Como ejemplo, seleccionaremos el proceso de inserción de imágenes en la nube. Que nos permitirá arrastrar o seleccionar el archivo de imagen que cargará en unos pocos segundos en nuestro sitio.

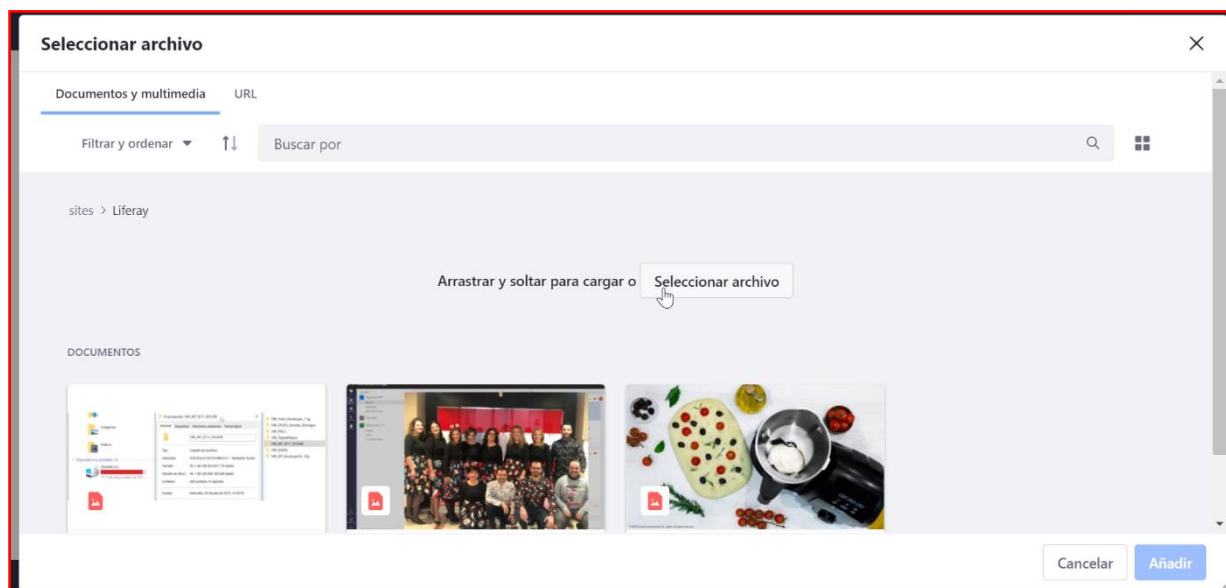


Ilustración 45: Ejemplo de inserción de imagen en la nube

Por otro lado, si definimos un editor de texto y no podemos hacerlo, para que serviría por lo que se muestra como sería.

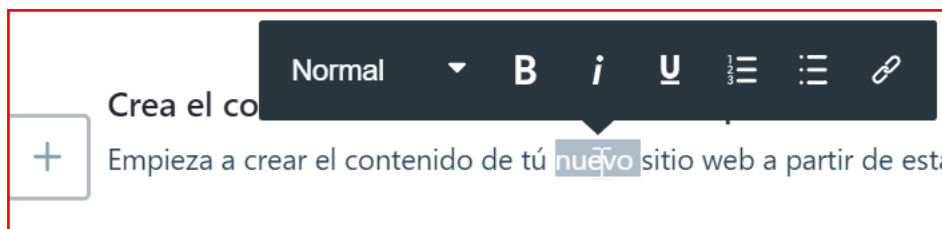


Ilustración 46: Edición de texto

Destacar que para la correcta maquetación se define que los procesos multimedia se sitúen dentro de cada línea, es decir el "+" situado a la izquierda. Por lo que, el editor de texto se sitúa en función de la situación del curso en el texto.

Finalmente, tendremos como resultado un código HTML que será almacenado por parte de Liferay en nuestra tabla y que en el momento de su generación final solo aplicará el identificador para así cargarla de forma dinámica.

6.1.3. Contenido multimedia en la nube

El proceso que hace que genere y guarde el contenido añadido en el punto anterior en la nube es el siguiente:

```

<auri:form enctype="multipart/form-data" action="<%= actionAddText %>" method="post" name="form">
  <div class="alloy-editor-container">
    <liferay-editor:editor
      contents="Empieza a crear el contenido de tú nuevo sitio web a partir de esta línea. Tranquilo no vas a romper nada..."
      cssClass="my-alloy-editor"
      editorName="alloyeditor"
      name="editor"
      placeholder="description"
      showSource="false" onChangeMethod="extractCodeFromEditor"/>
    </div>
    <div style="display: none">
      <input name="<portlet:namespace/>texto" type="hidden" value="" />
    </div>
  </auri:form>

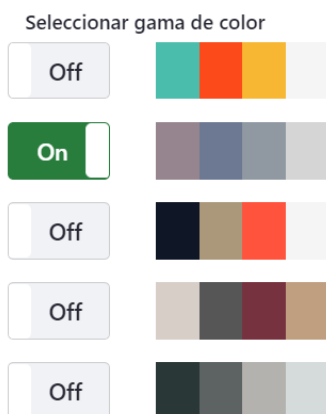
```

Ilustración 47: Construcción del servicio de edición de texto

6.1.4. Añadir/Editar menú

De nuevo, un pilar indispensable para la excelente experiencia del usuario, aunque parezca repetitivo, pero es importante. Ya que se centra el trabajo es un trabajo sencillo y robusto <simplemente observa el resultado> [11].

- Paleta de colores



Como se menciona en la fase de análisis el proceso de la paleta de colores es complejo pero sencillo al mismo tiempo [12], por lo que se genera una paleta de codificación de elementos entre padres e hijos, con lo que únicamente nuestro código solo cambia un número de la paleta.

```

<div class="row">
  <div class="entry input-group mb-3" id="paleta2">
    <input type="checkbox" data-toggle="toggle" data-onstyle="success" >
    <div class="c-1 canvas-1 canvas-all"></div>
    <div class="c-2 canvas-all"></div>
    <div class="c-3 canvas-all"></div>
    <div class="c-4 canvas-all"></div>
  </div>
</div>

```

Ilustración 48: Paleta de Color del menú

Ilustración 49: Código fuente de la paleta de selección

```

/*
 * INICIO PALETA DE COLORES
 */

div #paleta1 .c-1{
  background-color:#4ABDAC;
}
div #paleta1 .c-2{
  background-color:#FC4A1A;
}
div #paleta1 .c-3{
  background-color:#F7B733;
}
div #paleta1 .c-4{
  background-color:#F5F5F5;
}
div #paleta1 a.c-4t{
  color:#F5F5F5;
}
div #paleta1 a.c-5t{
  color:white; !important
}

div #paleta2 .c-1{
  background-color:#96858F;
}
div #paleta2 .c-2{
  background-color:#6D7993;
}

```

Como podemos ver en el pequeño fragmento de CSS y JS es simple lo demás es tener clara la estructura de inputs que formula el contenido, el cual se repite en las 3 columnas que forman el conjunto.

Ilustración 50: CSS de la paleta de Color del Menú

- Botones dinámicos

Una correcta gestión de los botones que forman parte de nuestra navbar y que se muestra en la parte superior ^[13] como guía para los usuarios es que se implemente un sistema de inputs dinámicos limitados a 5, los cuales se tiene previsto en futuras píldoras aumentas junto con la paleta de gamas de colores.

```
<div class="control-group" id="fields">
  <label class="control-Label" for="field1">Añadir botones al menú</label>
  <div class="controls">
    <form role="form" autocomplete="off">
      <div class="entry input-group mb-3">
        <input class="form-control" name="fields[]" type="text" placeholder="Inicio" />
        <span class="input-group-btn">
          <button class="btn btn-success btn-add" type="button">
            <span class="glyphicon glyphicon-plus"></span>
          </button>
        </span>
      </div>
    </form>
  </div>
</div>
```

```
cont=0;
$(function(){
  $(document).on('click', ".btn-add", function(e)
  {
    e.preventDefault();
    if(cont<4){
      var controlForm = $('<div class="controls form:first">'),
          currentEntry = $(this).parents('<div class="entry:first">'),
          newEntry = $(currentEntry.clone()).appendTo(controlForm);

      newEntry.find('input').val('');
      controlForm.find('<div class="entry:not(:last) .btn-add">')
        .removeClass('btn-add').addClass('btn-remove')
        .removeClass('btn-success').addClass('btn-danger')
        .html('<span class="glyphicon glyphicon-minus"></span>');
      cont++;
    }
  }).on('click', '.btn-remove', function(e)
  {
    $(this).parents('<div class="entry:first">').remove();
    cont--;
    e.preventDefault();
    return false;
  });
});
```

Ilustración 51: Código fuente creación botones menú

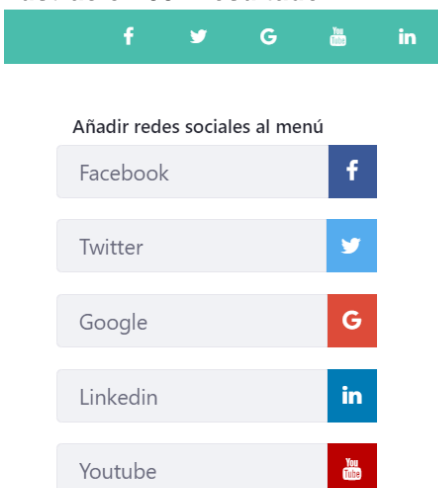


Ilustración 52: Resultado botones Menú^[15]

También se crea un método jQuery que nos ayudara a crear las líneas que elimina y añade los inputs. Teniendo en cuenta que ^[14] se gestionan de manera dinámica del lado del cliente.

- Redes sociales

Ilustración 53: Resultado



Por último, nos encontramos con la columna de redes sociales. La cual, seguirá el patrón de la anterior con la diferencia que añadirá los iconos de las redes sociales.

```
<div class="col-3 offset-md-1">
  <label class="control-Label" for="field1">Añadir redes sociales al menú</label>
  <div class="row">
    <div class="entry input-group mb-3">
      <input class="form-control" name="fields[]" type="text" placeholder="Facebook" />
      <span class="input-group-btn">
        <span class="fa fa-facebook"></span>
      </span>
    </div>
  </div>
</div>
```

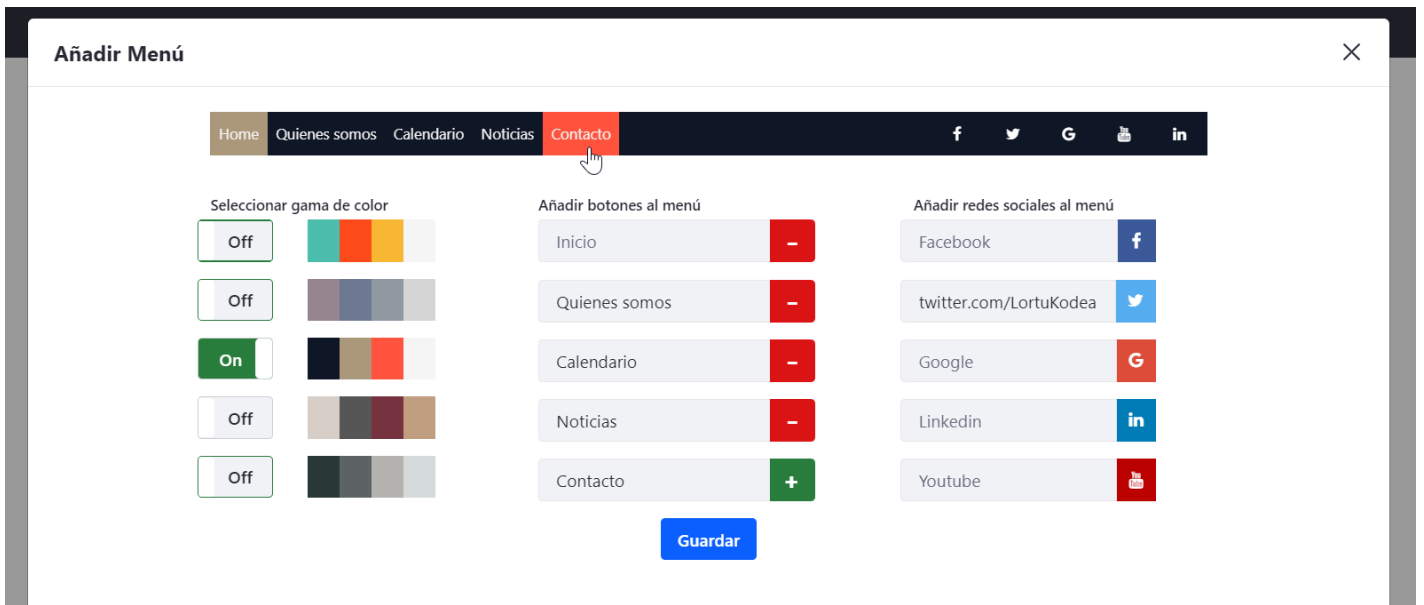


Ilustración 55: Resultado ventana de añadir Menú

6.1.5. Crear grupo y guardar

Durante el proceso de construcción de nuestra plantilla el usuario podrá guardar en cualquier momento su trabajo teniendo en cuenta la nueva funcionalidad de agrupación de plantillas, por lo que el diseño del apartado de guardado se rediseña para poder al mismo tiempo que podemos agrupar por grupos existentes por parte del usuario crearlos si no los hemos hecho. De esta forma:

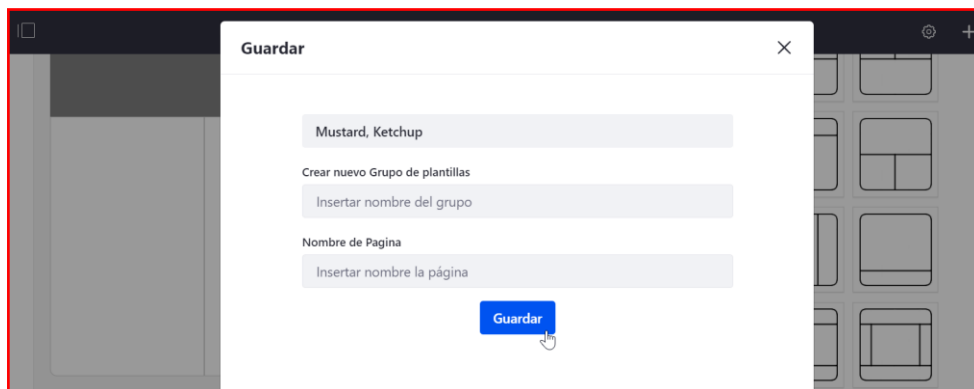


Ilustración 56: Ventana de guardado

Por lo tanto, hemos mejorado la funcionalidad y ampliado el proceso con el selector de grupos existentes durante el proceso de guardado.

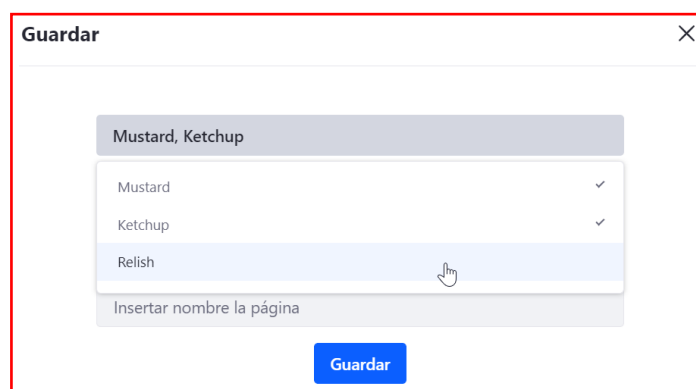


Ilustración 57: Selección de grupo ya existente

6.1.6. Comprimir y descargar

En el procesado de lectura de datos y generación de zips realizaremos una edición de un directorio de generación asociado a lid de usuario. Destacar que este paso es seguro y estable y no contempla consumo de memoria a largo plazo puesto que Liferay lo genera en temporales para su posterior eliminación una vez finalizado el proceso.

Por lo que nuestro compilador leerá los ficheros generados para su posterior tratamiento. Informar que se mostrara el método principal que genera el zip de descarga ya que el resto nos recursos como constantes y procesos de mejora y encapsulación que se pueden interpretar gracias a la correcta descripción del código siguiente.

```
/**
 * Función encargada de generar el ZIP para los ficheros de catálogos
 */

public static void crearZipCatalogo(){
    logger.info("+++ INICIO Proceso de lectura y guardado en zip de las plantillas +++");
    String dirRaiz = Constantes.RUTA_LIFEFS + Constantes.PLANTILLAS_HTML;
    logger.debug("La ruta padre donde se van a buscar los ficheros publicados es " +
dirRaiz);
    String nombreZipPlantillasPr =
Util.getNombreZipFichasProductosDate().concat("_160000");
    // creamos ficheros zip
    File ficheroZip = new File(Constantes.RUTA_LIFEFS + "tmp" + File.separator +
"PDF" +
        File.separator + nombreZipFichasPr.concat(".zip"));
    logger.debug("Creación inicial fichero zip "+ficheroZip.getName());
    // creamos un unico fichero de control.txt
    String rutaControl = Constantes.RUTA_LIFEFS + "tmp" + File.separator + "HTML"
        + File.separator + "control.txt";

    File ficheroControlFinal = new File(rutaControl);
    try {
        ficheroControlFinal.createNewFile();
    } catch (IOException e2) {
        logger.error("Error a la hora de crear nuevo fichero de "
+ficheroControlFinal.getName());
    }
    logger.debug("Creación fichero control.txt generico EN " +
ficheroControlFinal.getAbsolutePath());
    // .zip
    FileOutputStream fos = null;
    BufferedOutputStream bos = null;
    ZipOutputStream zos = null;
    try {
        fos = new FileOutputStream(ficheroZip);
        bos = new BufferedOutputStream(fos);
        zos = new ZipOutputStream(bos);
    } catch (FileNotFoundException e2) {
        logger.error("Fichero no encontrado. " +ficheroZip.getName());
    }
    // obtenemos fichero que hace de carpeta root
    File fileLifeFs = new File(dirRaiz);

    //preparamos las variables para recibir las plantillas
    List<File> listadoDocumentosIndividual = new ArrayList<File>();
    List<File> listadoFichasCorruptas = new ArrayList<File>();
    List<File> deleteFilesPubl = new ArrayList<File>();
    // obtenemos subdirectorios de la carpeta LiferayFS
    List<File> dirRoot = Arrays.asList(fileLifeFs.listFiles());
    List<File> subdir = new ArrayList<File>();
```

```

//Cargamos el contenido de los subdirectorios la PLANTILLA
for (File sub : dirRoot) {
    if(sub.getName().equalsIgnoreCase(Constants.Sitios_Web.LK.toString())){
        subdir.addAll(Arrays.asList(sub.listFiles()));
    }
}
//Montar zip
try {
    addToZipFile(ficheroControlFinal, zos);
    addListZipFile(listadoDocumentosIndividual, zos);
} catch (FileNotFoundException e1) {
    logger.error("Error fichero no encontrado a la hora de anyadir al zip el
fichero control "+e1);
} catch (IOException e1) {
    logger.error("Error a la hora de anyadir al zip el fichero control
"+e1);
}
// cerramos zip
try {
    zos.close();
} catch (IOException ex) {
    IOUtils.closeQuietly(zos);
} finally {
    IOUtils.closeQuietly(zos);
}
logger.info("cerramos zip de Plantilla y lo guardamos en la ruta:
"+ficheroZip.getAbsolutePath());
String rutaZipFichasDestino =
PropsUtil.get(PropsKeys.DL_STORE_FILE_SYSTEM_ROOT_DIR)
.concat(StringPool.FORWARD_SLASH).concat("PDF");
File carpetaDestinoZip = new File(rutaZipFichasDestino);
logger.info("Ruta destino ZipCatalogoFichas: " +
carpetaDestinoZip.getAbsolutePath());
carpetaDestinoZip.mkdirs();
try {
    FileUtils.copyFileToDirectory(ficheroZip, carpetaDestinoZip);
} catch (IOException e) {
    logger.error("Ruta destino zip catalogoPlantillas no encontrada.",e);
}
//Eliminamos las plantillas que no hemos podido tratar
Util.deleteListFichero(deleteFilesPubl);
//Eliminamos archivos temporales del tmp y fichero control generado
if (!ficheroControlFinal.delete() && !ficheroZip.delete()) {
    logger.error("Error intentando elimiar el fichero
"+ficheroControlFinal.getAbsolutePath());
} else {
    logger.info("Eliminado el fichero temporal:
"+ficheroControlFinal.getAbsolutePath()+" y el zip"+ ficheroZip.getAbsolutePath());
}
Util.borrarContenidoFichero(ficheroControlFinal);
logger.info("+++ FIN Proceso de lectura y guardado en zip de las fichasPDF de
Catalogo de productos +++");
}

```

6.2. Portlet: Visor de plantillas

En esté portlet vamos a poder visualizar y realizar diferentes acciones dentro del perfil del usuario siempre y cuando permanezca identificado pudiendo realizar ediciones sobre las diferentes plantillas generadas y guardadas como descargar y eliminar las mismas.

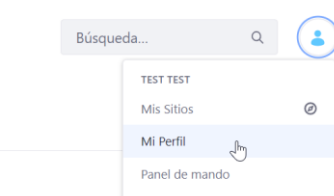


Ilustración 58: Perfil

Todo ello, enmarcado en una tabla para una mejor visualización y con diferentes botones de acción mencionados en el párrafo anterior. Por lo que, podemos ver en la siguiente imagen:

Show 10 entries Search:

Identificador	Usuario	Fecha	Estructura	Nombre Plantilla	SubRegistros
1	Test Test	2020-01-02 11:20:39.0			
101	Test Test	2020-01-02 12:57:02.0		prueba 2	
201	Test Test	2020-01-06 15:48:14.721			

Showing 1 to 3 of 3 entries Previous Next

Ilustración 59: Visor de plantillas del usuario

El visor de plantillas será una simple tabla construida con DataTables y que se genera automáticamente gracias a un foreach de jtl. Realizada previamente una petición a la API de la siguiente forma:

```
public List<Estructura> findByUserId(long groupId){
    return estructuralLocalService.findByGroupId(groupId);
}
```

Ilustración 60: Ejemplo de petición a la API del Visor de plantillas

Por lo que, ya en el portlet de visualización llamaremos a la API para obtener las plantillas del usuario actual:

```
public class LKVisorPlantillasPortlet extends MVCPortlet {

    private static Logger _logger;

    @Override
    public void doView(RenderRequest renderRequest, RenderResponse renderResponse)
        throws IOException, PortletException {

        _logger = LoggerFactory.getLogger(this.getClass().getName());

        ThemeDisplay themeDisplay= (ThemeDisplay) renderRequest.getAttribute(WebKeys.THEME_DISPLAY);
        long userId= themeDisplay.getUserId();
        List<Registro> listRegistrosByUser = RegistroLocalServiceUtil.listRegistrosByUserId(userId);
        renderRequest.setAttribute("listRegistrosByUser", listRegistrosByUser);

        super.doView(renderRequest, renderResponse);
    }
}
```

Ilustración 61: Código fuente que carga el contenido del Visor de plantillas

Por tanto, en el apartado del cliente tendremos que realizar la implementación para la generación automática de toda la tabla:

```

<table id="example" class="display" style="width:100%">
  <thead>
    <tr>
      <th>Identificador</th>
      <th>Usuario</th>
      <th>Fecha</th>
      <th>Estructura</th>
      <th>Nombre Plantilla</th>
      <th>SubRegistros</th>
      <th>&nbsp;</th>
    </tr>
  </thead>
  <tbody>
<c:forEach var='r' items='${listRegistrosByUser}' >
  <tr>
    <td>${r.registroId}</td>
    <td>${r.userName}</td>
    <td>${r.modifiedDate}</td>
    <td>
      <CENTER>
        
      </CENTER>
    </td>
    <td>${r.nameRegistro}</td>
    <td>${r.subRegistos}</td>
    <td>
      <jsp:include page="actions.jsp" >
        <jsp:param value="${r.registroId}" name="registroID" />
        <jsp:param value="${r.estructuraId}" name="estructuraID" />
      </jsp:include>
    </td>
  </tr>
</c:forEach>
</table>

```

Ilustración 62: Código del Visor de plantillas del Usuario

6.2.1 Acciones

Dentro de la generación de la tabla podemos ver como se incluye la sitio de acciones donde se incluyen las diferentes acciones que se podrán realizar sobre cada registro.

```

<portlet:renderURL var='principal'>
  <portlet:param name="jspPage" value="/principal.jsp" />
  <liferay-portlet:param name="estructuraID" value="${r.estructuraId}" />
  <liferay-portlet:param name="registroID" value="${r.registroId}" />
</portlet:renderURL>

<CENTER>
  <button type="button" class="btn btn-primary" title="Editar" onclick="Location.href='<%=principal%>';" ><i class="fa fa-edit"></i></button>
  <button type="button" class="btn btn-success" title="Descargar"><i class="fa fa-arrow-circle-down"></i></button>
  <button type="button" class="btn btn-danger" title="Eliminar"><i class="material-icons" style="font-size:18px">&#xe872;</i></button>
</CENTER>

```

Ilustración 63: Código de los botones de acción

6.3. Invitar a miembros

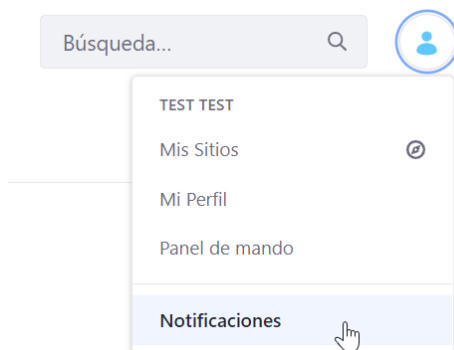


Ilustración 64: Notificaciones

Como parte del proceso de expansión se implementa un portlet para el envío de correos electrónicos a conocidos u otros miembros para compartir contenidos.

El usuario recibirá un correo con el enlace para registrarse en el portal. Estas acciones contendrán beneficios para los usuarios en futuras pildoras.

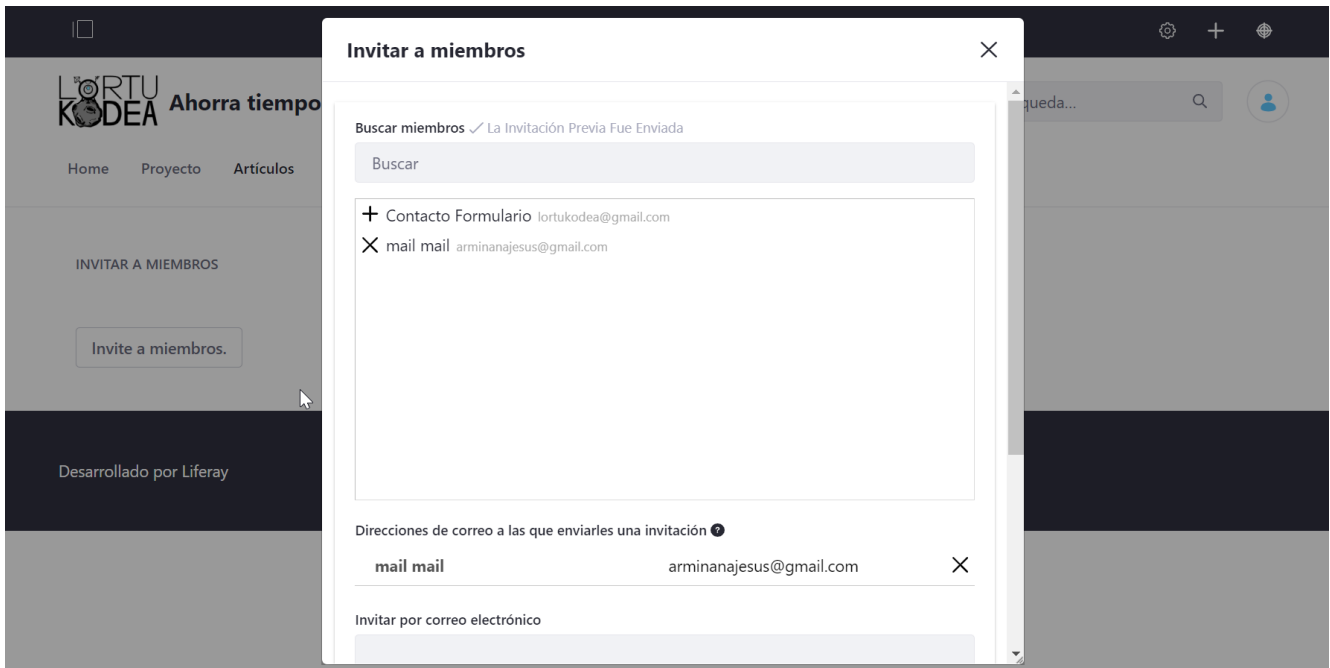


Ilustración 65: Ventana invitar a miembros

6.4. Blog de Noticias

En el apartado del menú principal del portal se incluye dentro de Proyecto un blog con información relevante sobre el proyecto y su avance. Pudiendo ser de libre acceso por parte de todos los usuarios editado por los administradores y redactores, con la presencia de diferentes acciones como comentarios, valoraciones o vincular comentarios y publicaciones en redes sociales.



Ilustración 66: Blog

7. Manual de Usuario

7.1. Registro

Cuando el usuario entre por primera vez en el sitio tendrá que registrar para poder acceder a las herramientas de la aplicación:

The screenshot shows the registration page for 'LORTU KODEA'. The header includes the logo and the slogan 'Ahorra tiempo y trabajo!'. A search bar and a login link are in the top right. The main content area is titled 'CREAR CUENTA' and contains a registration form with the following fields:

- Nombre de usuario *: test@liferay.com
- Contraseña *: [Redacted]
- Confirmación *: [Redacted]
- Dirección de correo *: [Redacted]
- Fecha de nacimiento: 01/01/1970
- Lenguaje: español (España)
- Prefijo: [Redacted]
- Nombre *: [Redacted]
- Apellido *: [Redacted]
- Texto de verificación *: [Redacted]

A 'Guardar' button is located at the bottom left of the form area.

Ilustración 67: Formulario de alta de usuarios

Si todo fue correctamente recibirá un correo electrónico validando su cuenta de correo electrónico y ya podrá acceder al portal.

The screenshot shows a modal window titled 'Acceder'. It contains the following elements:

- Dirección de correo: test@liferay.com
- Contraseña: [Redacted]
- Recuérdame
- Acceder button
- Links: + Crear cuenta and He olvidado mi contraseña

Ilustración 68: Acceso usuarios registrados

7.2. Crea mí primera plantilla

Una vez, en la página principal el usuario podrá crear plantillas, un total de 856 combinaciones diferentes únicamente en combinaciones de estructuras de plantillas.

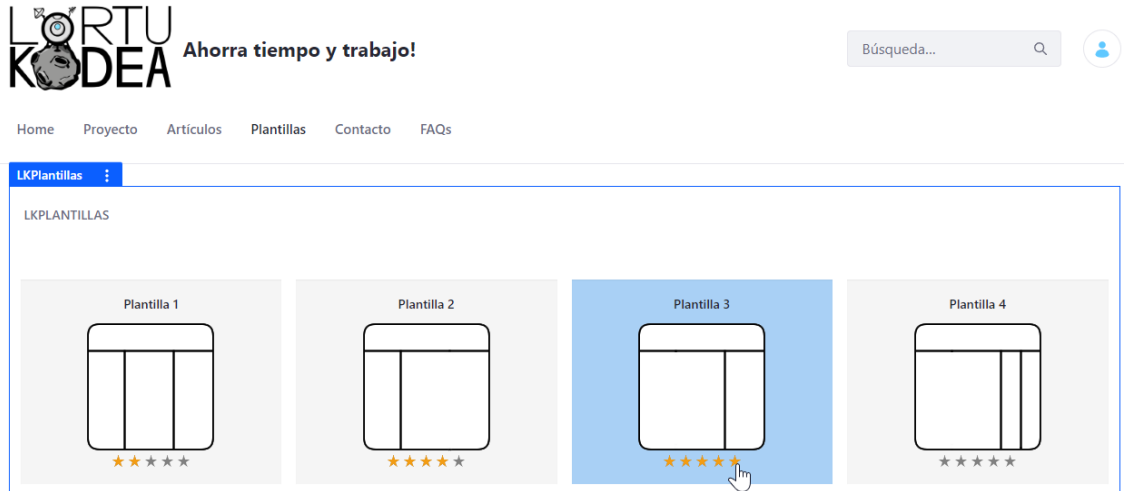


Ilustración 69: Ejemplo de selección de plantilla principal

7.3. Administrar mí plantilla

Una vez, en la pantalla de edición de plantillas podrá realizar diferentes acciones mencionadas en los apartados de implementación y desarrollo.

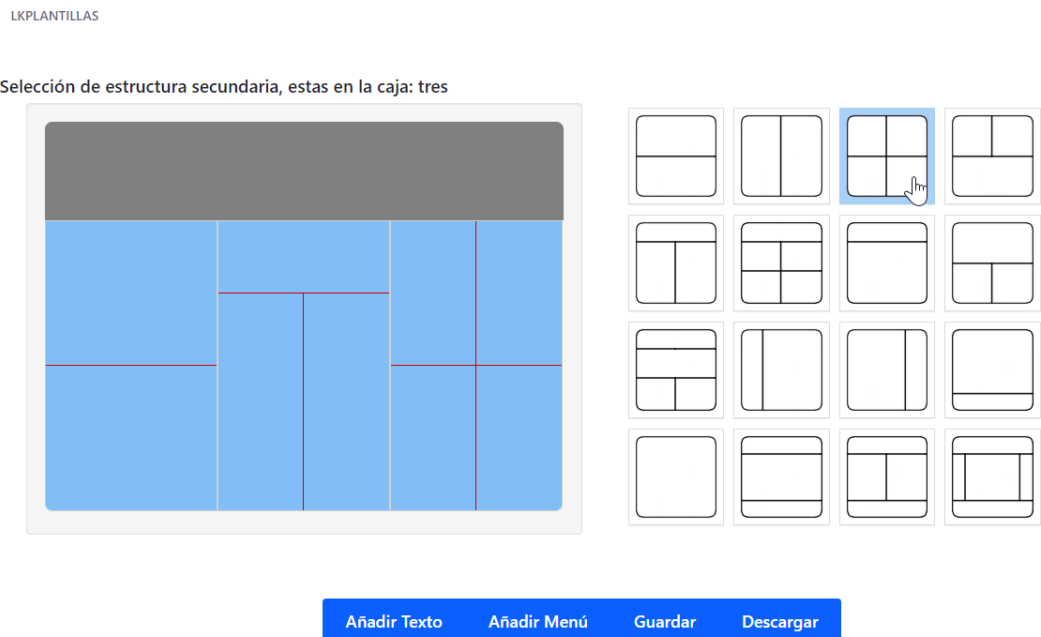


Ilustración 70: Ejemplo de edición de estructura de plantilla

8. Conclusiones

✓ Primera fase:

- Cambios en la versión de Java vinculadas las S.O Windows 10 PRO. Se realizó un cambio en la variable de entorno del JAVA_HOME para apuntar a la versión jre del JDK de Java para una correcta estabilidad del entorno ^[10] de desarrollo.
- Por facilidad se descarga todo el paquete de desarrollo del Developer Studio de Liferay para evitar conflictos con el Eclipse ya instalado.
- Se identifican diversos problemas/bugs de liferay que son corregidos en el proceso inicial del desarrollo del workspace del proyecto ^[10] (Creación y configuración).
- En cada proceso de compilación de dependencias se detecta que se producen errores de dependencias por culpa de ciertos criterios de configuración de seguridad estrictica, tanto en la red como en la configuración ^[10] del sistema operativo.

✓ Segunda fase:

- Memoria final: Se remite la redacción del manual de usuario, presentación en PowerPoint y video (demo) para la entrega final, junto con el código fuente.
- Para la entrega final se implementará un visor de plantillas en el panel de usuario.
- Durante el proceso de desarrollo de la segunda fase se han encontrado diferentes inconvenientes entre los cuales, podemos destacar:

- Problemas en el redesplicue de las modificaciones en la recopilación de los servicios que provocó que al no tener una sincronización entre todas las partes producía una incongruencia entre el front y el back de la aplicación Liferay que eludía desplegando automáticamente el último “.war” ^[10] compilado sin problemas.
- Problemas de versionado entre librerías JS necesarias para las diferentes acciones y funcionalidades presentes en cada una de ellas y que han requerido la adquisición de una versión que englobe todas ellas para una mejor experiencia ^[9] de usuario.

9. Glosario

- **Portlet:** componente modular, similar a una aplicación que se despliega en un portal.
- **Liferay Marketplace:** catálogo de plugins para Liferay Portal accesible por web y a través del panel de control de Liferay. Lista distintos plugins desarrollados tanto por Liferay Inc como por desarrolladores o empresas no vinculadas a Liferay y permite instalarlos en nuestro portal.
- **Service Builder:** herramienta de generación de código mediante mapeo objeto-relacional (ORM) que genera las clases necesarias para la interacción de un portlet con una base de datos a partir de la definición de entidades en XML.
- **OSGi:** es un conjunto de estándares para la construcción de sistemas modulares. Anteriormente era difícil de aprender y usar, pero la aparición de servicios declarativos, ha facilitado mucho el aprendizaje y el uso de los mismos.
- **API:** se proporcionan a través de portal-kernel (anteriormente conocido como portal-service); Todas las demás API públicas son proporcionadas por sus propios módulos.
- **Integración completa:** herramientas específicas de Liferay (como Service Builder) dentro de Maven y Gradle. Además hemos adoptado algunas nuevas herramientas como Bnd.
- **Aislar funcionalidad:** Es natural poner el foco en desarrollar una componente de software a la vez. En un módulo, trabajamos en un pequeño conjunto de clases para definir e implementar la función del módulo. Mantener el componente pequeño facilita la escritura de código elegante y de alta calidad.

Cuanto más coherente sea el código, más fácil será probar, depurar y mantener. Los módulos se pueden combinar para proporcionar una nueva función, independientemente de la función de cada módulo.

- **Encapsulado:** Un módulo encapsula una función (capacidad). Las implementaciones de módulos están ocultas para los consumidores, por lo que puede crearlas y modificarlas a su gusto.

A lo largo de la vida útil de un módulo, puede corregir y mejorar la implementación o sustituirlo completamente. Usted hace los cambios, totalmente transparente para los consumidores. El contrato de un módulo define su capacidad e interfaz, haciendo que el módulo sea fácil de entender y usar.

10. Bibliografía

- [1]. <https://liferay.dev>
- [2]. <https://liferay.dev/forums>
- [3]. <https://github.com>
- [4]. <https://www.albertcoronado.com/Tots/liferay>
- [5]. <https://www.apuntesdejava.com>
- [6]. <https://www.draw.io>
- [7]. <https://getbootstrap.com>
- [8]. <https://alloyui.com>
- [9]. <https://stackoverflow.com>
- [10]. <https://www.w3schools.com>
- [11]. <https://codepen.io>
- [12]. <https://www.liferaystack.com>
- [13]. <http://www.liferaysavvy.com>
- [14]. <https://www.obelearningservices.com>
- [15]. <https://bootsnipp.com>
- [16]. <https://www.asesoriaensig.com.mx>
- [17]. <http://www.liferaysavvy.com>
- [18]. <https://issues.liferay.com>
- [19]. <https://datatables.net>

11. Anexos

Anexo A: Liferay: conceptos básicos

Durante mucho tiempo Liferay Portal se ha definido como una plataforma para la gestión de portales OpenSource.

- ▶ A partir de la versión 6.1, Liferay Portal es una plataforma web corporativa que le ayudará a desarrollar soluciones empresariales con resultados inmediatos y valor a largo plazo.
- ▶ Liferay Portal es más que un portal, es una plataforma de integración.
- ▶ Esta plataforma proporciona una solución interesante para el sector público y el sector privado.
- ▶ Liferay Portal es una plataforma web que nos permite crear, gestionar y explotar diferentes aplicaciones web con un nivel de personalización muy elevado, de forma fácil y muy simple.
- ▶ Las aplicaciones web que construye se consumen desde todos los clientes: desktop, mobile, o cualquier cosa intermedia.
- ▶ Proporciona todas las aplicaciones estándares necesarias para construir sitios web y proporciona un framework de desarrollo fácil de utilizar para crear o personalizar aplicaciones nuevas.
- ▶ Además, Liferay Portal ha sido desarrollado usando una estrategia open source gracias a personas en todo el mundo. El código base es muy sólido y ha sido probado en entornos críticos.

Por dónde empezar con Liferay Digital Experience Platform



Servicios Digitales

Portales de autogestión para clientes, franquiciados, agentes, etc.



Experiencias Web

Sitios web públicos que ofrecen experiencias personalizadas a clientes y usuarios anónimos.



Entorno de Trabajo Digital

Intranets modernas que conectan equipos y fomentan la colaboración.



Experiencias Móviles

Experiencias omnicanal a través de cualquier dispositivo.

Construido sobre el portal open source líder en el mundo.

FORRESTER

Conoce por qué Forrester cree que las tecnologías de portal son cruciales para cualquier estrategia de experiencia digital.

[Lee el "Vendor Landscape" >](#)

Gartner

Liferay alcanza máximas puntuaciones frente a 16 proveedores para casos de uso comunes de portal.

[Lee el Informe de Gartner >](#)

Ser un líder en tu sector



Banca

Poner a los clientes en primer lugar con servicios de banca digital mejorados.



Sanidad

Capacitar a los pacientes, ofreciéndoles información y herramientas para el bienestar, prevención y cuidado de su salud.



Administración Pública

Maximizar los presupuestos ofreciendo servicios digitales, sencillos de utilizar.



Retail

Personalizar tus tiendas con promociones específicas y servicios mejorados digitalmente.



Automoción

Transformar la experiencia en el concesionario con un enfoque que ponga al cliente en primer lugar.



Educación

Experiencias de aprendizaje digitales que se ajusten a las necesidades de cada estudiante.

Liferay Portal ofrece un conjunto de capacidades muy interesantes:

- ▶ Gestión de contenidos y documentos.
- ▶ Publicación web y espacios de trabajo compartidos.
- ▶ Entornos colaborativos.
- ▶ Portales empresariales.

Sus características más importantes son:

- ▶ Trabaja sobre la mayoría de los servidores de aplicaciones y contenedores web, bases de datos y sistemas operativos.
- ▶ Trabaja con las últimas tecnologías Java.
- ▶ Usa un framework SOA abierto.
- ▶ Cumple las especificaciones de portlets JSR-168 y JSR-286.

- ▶ Proporciona en torno a 60 portlets para el uso en los portales.
- ▶ Soporte a i18n.
- ▶ Panel de control muy completo que permite manejar la configuración del sistema de forma cómoda.

Sus características más importantes son (cont.):

- ▶ CMS: sistema de gestión de contenidos.
- ▶ WCM: gestor de contenidos web.
- ▶ Soporte a redes sociales gracias a un conjunto de portlets.

Amplia documentación:

The banner for the Liferay Developer Network includes the Liferay logo and the text "DEVELOPER NETWORK". It features a search bar with the word "Buscar" and a button labeled "Acceder". Below the banner, there are four main categories:

- USER & ADMIN:** Build your web site, collaborate with your colleagues, manage your content, and more.
- DEVELOPER:** Build applications that run inside Liferay, extend the features provided out of the box with Liferay's APIs.
- MARKETPLACE:** Let the world know about your app by publishing it in Liferay's marketplace.
- COMMUNITY:** Help build the Liferaypedia and soon answer questions in the forums, spread the word about Liferay.

Marketplace:

The screenshot shows the Liferay Marketplace interface. At the top, there is a header with the "Marketplace" logo and navigation links for "Categorías", "Overview", "Developers", and "Descargar Plugin para Marketplace". Below the header, there is a "Featured Apps" section. The first part of this section shows a "LIFERAY MANAGER APP CONTEST! Featured Apps" banner with icons for "Social Apps Proxy", "Visioneo reports PE", "Tori Forum", and "Vahtie EE - Survey and Insight". Below this, there is a search bar and filters for "Categoría", "Versión de Liferay", and "Precio". The main content area displays a grid of app cards, each with an icon, name, developer, and price:

App Name	Developer	Productivity	Price
Visioneo reports PE	Dominique Pardon	Productivity	€2,450,00 EUR
Vahtie EE - Survey and Insight	Arcusys Ltd.	Productivity	€22,000,00 EUR
Social Apps Proxy	Stian Sigvartsen	Communication	Gratis
Tori Forum	Vaadin Ltd	Communication	Gratis
TeamWorXX	PFI Knowledge Solutions Ltd	Productivity	€3,200,00 EUR

Anexo B: Instalación y configuración BBDD

- Descarga de la plataforma.

<https://web.liferay.com/es/digital-experience-platform/30-day-trial>

To get started on your 30-day trial of Liferay DXP:

1. Click on the "Download " button below for your 30-Day Trial.
2. Check your e-mail for your trial key*. If you did not receive an e-mail, please ensure that your e-mail is correctly entered on your Account Page.
3. Place the activation key into the Liferay "deploy " directory and start your application server.

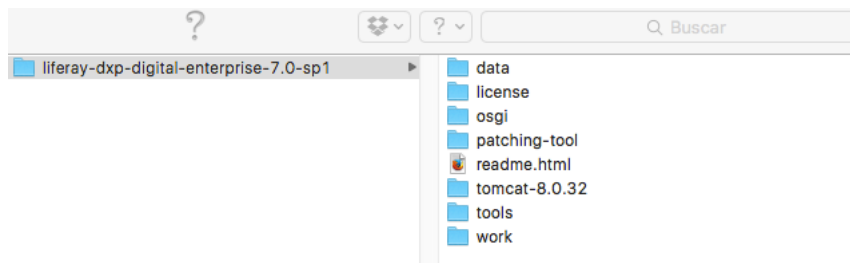
**A "trial key " is restricted to a limited number of connections.*

- CE vs EE.

Liferay Inc. ofrece dos ediciones de Liferay Portal: Una versión empresarial y otra de la comunidad.

- La versión empresarial.

Se descarga un fichero en formato zip. Al descomprimir encontramos una estructura de directorios, para el bundle de tomcat sería tal que así:



El correcto funcionamiento de una versión empresarial implica el despliegue de una clave de licencia:

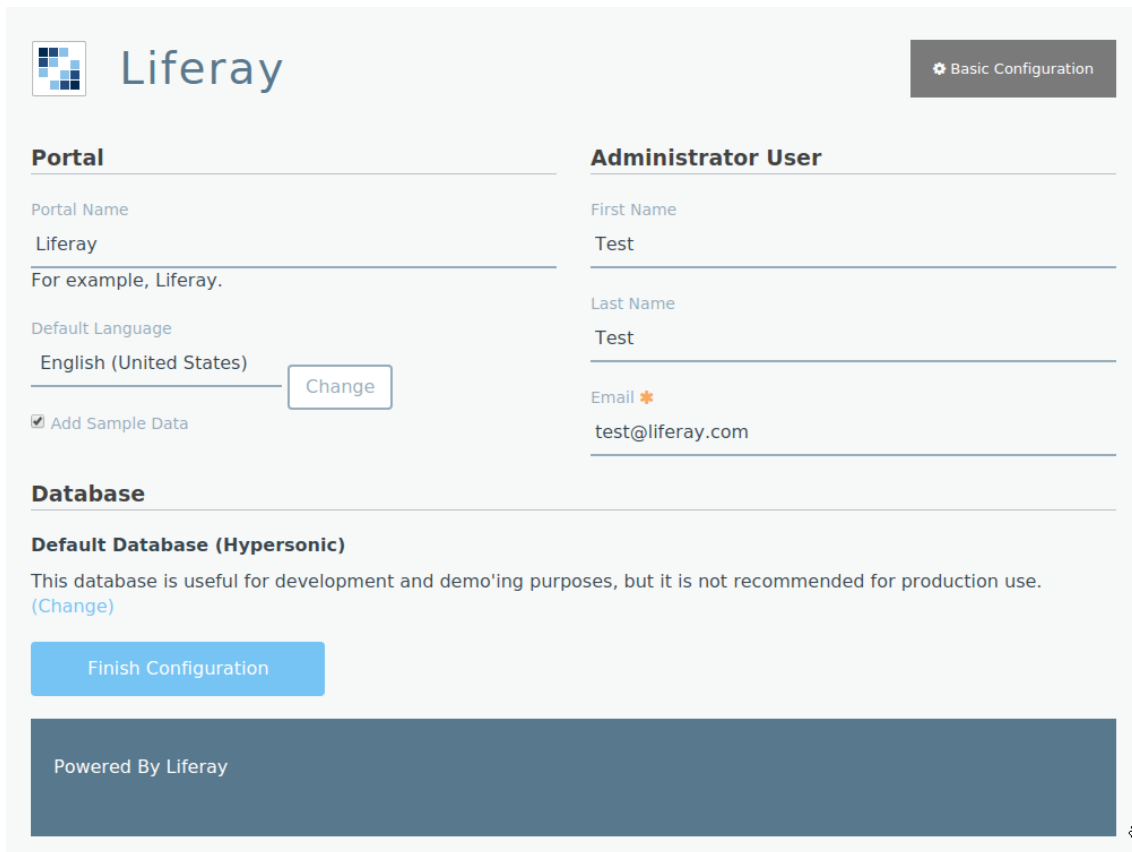
- ▶ Obtención de clave de licencia enviada por Liferay.
- ▶ Copiar clave de licencia en carpeta deploy en la estructura de directorios de Liferay Portal.

Arrancar el sistema (para Tomcat):

```
Window  
  
> startup  
  
Unix / Linux / Mac  
  
> ./startup.sh
```

```
En el navegador  
  
http://IP_MAQUINA:PUERTO  
  
Por ejemplo:  
http://localhost:8080
```

Configuración en arranque:



The screenshot shows the Liferay Basic Configuration page. At the top left is the Liferay logo. At the top right is a 'Basic Configuration' button. The page is divided into three main sections: Portal, Administrator User, and Database. The Portal section includes fields for Portal Name (Liferay), Default Language (English (United States)), and a checkbox for Add Sample Data. The Administrator User section includes fields for First Name (Test), Last Name (Test), and Email (test@liferay.com). The Database section includes a 'Default Database (Hypersonic)' section with a note that it is for development and demo purposes. A 'Finish Configuration' button is located at the bottom of the configuration area. A 'Powered By Liferay' footer is at the bottom of the page.

Portal

Portal Name
Liferay
For example, Liferay.

Default Language
English (United States) [Change](#)

Add Sample Data

Administrator User

First Name
Test

Last Name
Test

Email ✨
test@liferay.com

Database

Default Database (Hypersonic)
This database is useful for development and demo'ing purposes, but it is not recommended for production use.
([Change](#))

[Finish Configuration](#)

Powered By Liferay

Liferay de un vistazo:

Cuando Liferay está instalado, uno puede comenzar a configurarlo adaptándolo a su proyecto en particular.

- Gran cantidad de operaciones se podrán llevar a cabo mediante la interfaz gráfica de Liferay.

La autenticación se realiza haciendo click en la esquina superior derecha, sobre el enlace Acceder:



The screenshot shows the Liferay login form, titled 'Acceder'. It has a close button (X) in the top right corner. The form contains the following elements: a 'Dirección de correo' field with the text '@liferay.com' entered; a 'Contraseña' field; a checkbox labeled 'Recuérdame'; a green 'Acceder' button; and two links at the bottom: '+ Crear cuenta' and 'He olvidado mi contraseña'.

Acceder

Dirección de correo
@liferay.com

Contraseña

Recuérdame

Acceder

+ Crear cuenta He olvidado mi contraseña

Una vez autenticado, aparece una barra superior. Por defecto, nos ofrece las funcionalidades del gestor:

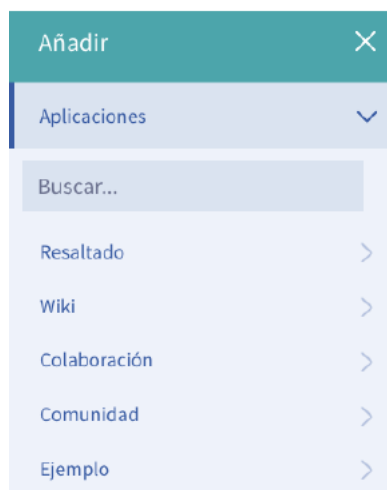
Esquina superior derecha:

- ▶ Configurar página.
- ▶ Añadir.
- ▶ Simulación.
- ▶ Sección de usuario.

Esquina superior izquierda: Menú.

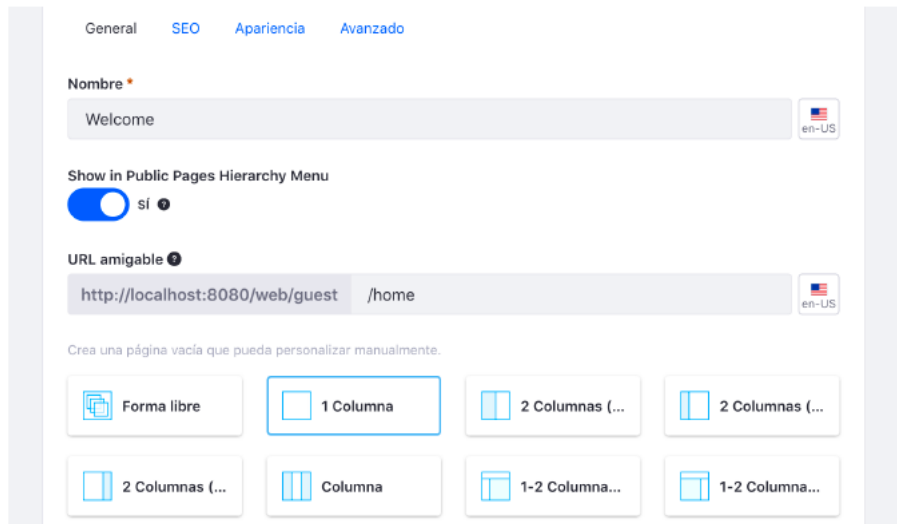


Añadir: Este enlace nos permite incorporar aplicaciones y contenidos en las páginas del portal.

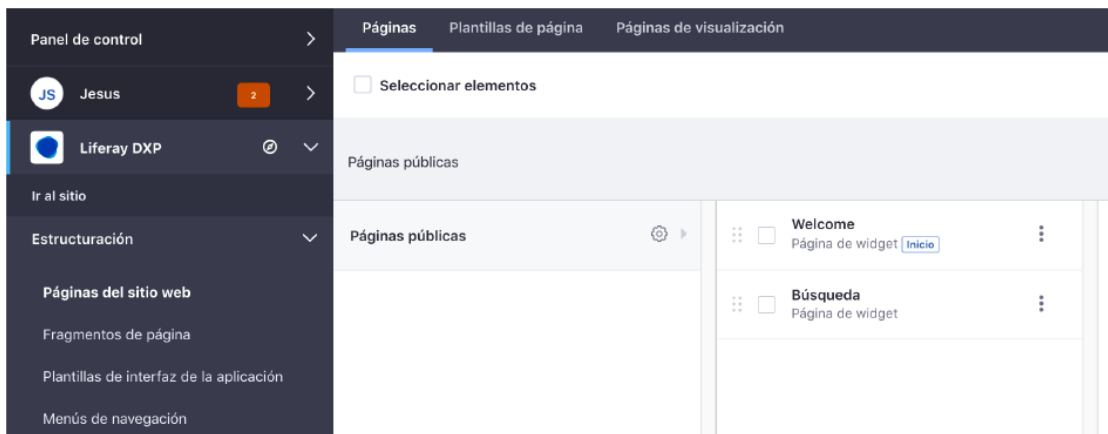


Configuración de la página:

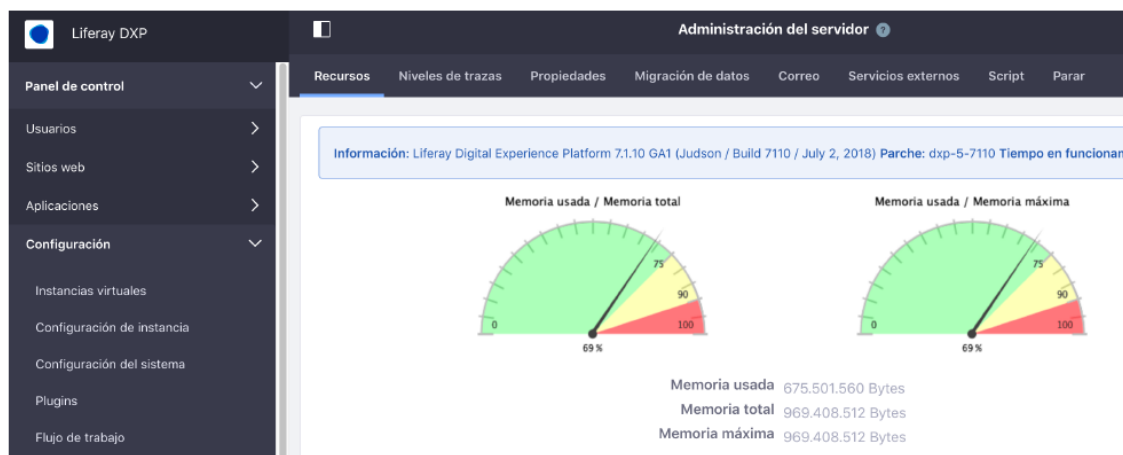
Esta opción permite cambiar la plantilla de la página, seleccionando alguna de las que propone.



Administrar páginas:



Panel de control: el menú nos ofrece un acceso al panel de control.



¿Qué es un portal en Liferay?

- ▶ Es una aplicación web que proporciona personalización, autenticación, mecanismos para incorporar contenidos en una página web y centraliza la capa de presentación de un sistema de información.
- ▶ Un portal debe tener características avanzadas para la personalización de los contenidos de usuario.
- ▶ Las páginas de un portal pueden estar formadas por los contenidos generados por diferentes componentes (por ejemplo, portlets).

Ejemplo: goodwill.org

- ▶ Es una de las organizaciones no gubernamentales más grandes a nivel mundial, que proporciona servicios profesionales, educación y entrenamiento a discapacitados, parados de larga duración, para ayudar a su incorporación al mercado de trabajo.
- ▶ Principales Funcionalidades: Chats colaborativos, foros, blogs, tableros de anuncios, elearning, librería documental.

Todas las características de un portal se apoyan en el siguiente concepto:

- ▶ Integrar diferentes aplicaciones en la misma pantalla.
- ▶ El usuario interactúa con el portal y el portal interactúa con diferentes aplicaciones de backend, integrándolas a todas en una única ventana del navegador sobre la que el usuario actúa.
- ▶ Esta integración habitualmente se conoce como at the glass.

Analicemos las características más importantes de la definición de portal:

- ▶ Personalización de usuario: permite que el usuario personalice los contenidos que ve en las páginas del portal.
- ▶ Personalización del backend: permite al administrador personalizar un determinado componente para un entorno específico.
- ▶ Single sign on: el usuario tendrá que autenticarse una sola vez.

Incorporación de contenido:

- ▶ Concepto de componente: permite que se desarrollen componentes que se puedan incorporar en el portal. Son como bloques de construcción.
- ▶ Los portales son, cada vez más, productos muy extendidos en el mercado. Se aplican sobre diferentes segmentos y usuarios del mercado.
- ▶ Esta variedad lo hace algo confuso. Cuando se quieren diseñar aplicaciones para portales se debe tener la cuenta que cada entorno exige requisitos diferentes.

Existe una clasificación tradicional:

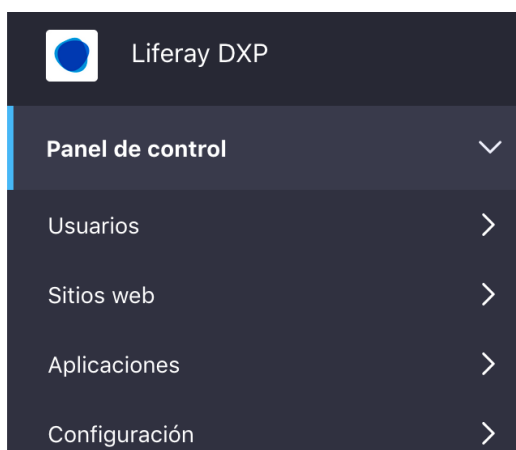
- ▶ Portales verticales y horizontales:
 - ▶ En los inicios del desarrollo de portales, éstos se construían enfocados sobre tareas y grupo de usuarios.
 - ▶ Las clasificaciones tradicionales se basaban en la relación entre los usuarios y el host del portal:
 - ▶ B2C: soporte de los clientes de una empresa específica. Las funcionalidades habituales son disponibilidad de productos, gestión de pedidos, información de los productos, etc. Esos portales suelen ser el front-end de un CRM (Customer Relationship Management).
 - ▶ B2B: relaciones entre diferentes empresas y la empresa que mantiene el portal. Estos portales habitualmente actúan como front-end de SCM (Supply-Chain-Management) ofreciendo funciones como órdenes de compra, facturas, etc.
 - ▶ B2E: portales intranet que proporcionan a los empleados de la empresa un acceso unificado a los sistemas (noticias de la empresa, motores de búsqueda, grupos de discusión, etc.).
 - ▶ Internet: portal accesible por cualquiera, tales como, Yahoo, Google, ... que ofrecen servicios como emails, noticias, motor de búsqueda.

Estas categorías empezaron a desaparecer dado que un usuario podía ser un empleado en un momento y un consumidor al siguiente.

- ▶ Los usuarios no deberían tener que aprender diferentes formas de relacionarse con un portal en función del role que tienen asociado en cada momento.
 - ▶ En ese momento aparece el concepto de portal vertical y horizontal.
 - ▶ La visión tradicional de un portal representa al portal vertical, centrado en una aplicación específica (CRM, SCM, etc.).
 - ▶ En cambio, los portales horizontales son portales integradores. Agregan contenidos de portales en nuevos portales. Eso permite al usuario final interactuar con un único portal.
- ▶ Portales macro, micro y nano:
- ▶ macro: son portales que integran aplicaciones de back-end existentes asociadas a una interface gráfica sólida. El número de usuarios que acceden a este tipo de portal es muy grande.
 - ▶ micro: se encuentran una escala por debajo de los macro portales. Habitualmente son usados por un único usuario o un grupo de usuarios pequeño.
 - ▶ nano: se encuentra en nivel más bajo, se integran en dispositivos electrónicos y actúan como interfaz de usuario para esos dispositivos.

Si eres administrador el panel de control tiene cuatro secciones:

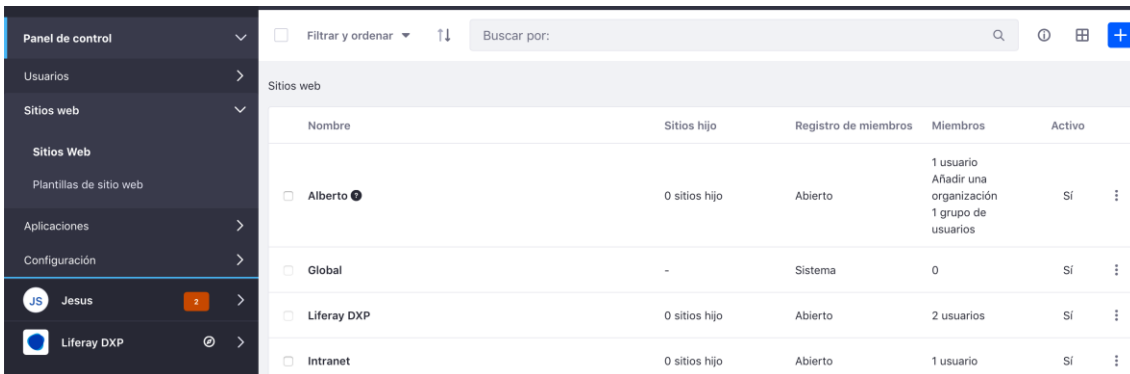
- ▶ Usuarios, Sitios web, Aplicaciones, Configuración.



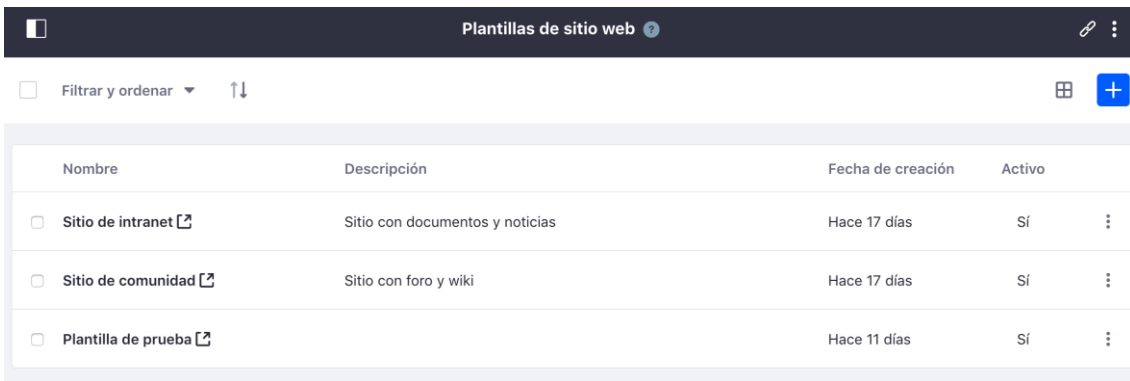
Esta sección nos permite gestionar usuarios, organizaciones, grupos de usuarios, roles, etc:



Esta sección nos permite gestionar el concepto de sitio web dentro de Liferay Portal:



Desde aquí se gestionan las plantillas de sitio:



Estas plantillas nos permiten crear sitios web de una forma muy rápida. Nos permite gestionar las aplicaciones que se ejecutan dentro de la plataforma:



Gestor de aplicaciones:

The screenshot shows the 'Aplicaciones' (Applications) section of the Liferay DXP administration console. At the top, there is a search bar labeled 'Buscar por:' and a filter/order dropdown. Below this, the main content area is titled 'Gestor de aplicaciones' and contains a list of installed and available applications. Each application entry includes a thumbnail, the application name, a brief description, version number, and status.

Nombre	Versión	Estado
Ecommerce Theme	1.0.0.0	Desinstalado
Independent Modules	1.0.5	Activo
Liferay Adaptive Media	2.0.5	Activo
Liferay Alloy	1.0.3	Activo

Nos permite configurar el comportamiento de la plataforma, por ejemplo: Instancias virtuales, configuración general, Liferay Sync mecanismos de autenticación y componentes.

The screenshot displays the 'Configuración de instancia' (Instance Configuration) page in the Liferay DXP administration console. The left sidebar shows a navigation menu with options like 'Panel de control', 'Usuarios', 'Sitios web', 'Aplicaciones', and 'Configuración'. The main content area is titled 'Configuración de instancia' and is divided into several sections: 'General', 'Configuración principal', 'Servidor virtual', and 'Navegación'. Each section contains input fields for various configuration parameters.

Sección	Parámetro	Valor
Configuración principal	Nombre *	Liferay DXP
	Host HTTP del CDN	
Configuración principal	Dominio de correo *	liferay.com
	Host HTTPS del CDN	
Servidor virtual *	Servidor virtual *	localhost
	Habilitar recursos dinámicos de CDN	<input checked="" type="checkbox"/>
Navegación	URL de Inicio	
	Página de salida por defecto	
	Página de entrada por defecto	