



GRADO EN INGENIERÍA INFORMÁTICA

TRABAJO FINAL DE GRADO

Portal web Ayuntamiento de Vinaros

Autor:
Alex BUENO AVANTE

Supervisor:
Cesar MONES
Tutor académico:
Lledó MUSEROS CABEDO

Fecha de lectura: 16 de Noviembre de 2015
Curso académico 2014/2015

Resumen

El siguiente proyecto consiste en el recorrido de las distintas fases que se requieren para desarrollar un portal web. Un portal web que ha de ser capaz de mostrar una imagen actualizada acorde con las tecnologías actuales. Se espera que ofrezca herramientas de participación para los ciudadanos interesados en aportar su opinión en las tareas de gobernación. El portal ha de facilitar la labor de los trabajadores de la corporación en su empeño de mostrar total transparencia a la hora de gestionar recursos municipales. También se espera agrupar los distintos portales propiedad del ayuntamiento bajo una jerarquía definida y controlada. Esto propiciará forjar y afianzar la imagen de la corporación. La mayoría de características vienen ofrecidas por el gestor de contenidos Liferay. Se analiza cómo se adapta el gestor a las necesidades y se implementa una pequeña funcionalidad para entender mejor el alcance de este.

Se observan las fases iniciales de planificación, donde se intenta predecir con el máximo acierto el transcurso y coste de las tareas.

Coordinar y gestionar las fases de análisis es la tarea que más tiempo absorbe y en la cual se destina el mayor esfuerzo. Gracias a esto, se obtiene una documentación lo suficiente precisa para los intereses del ayuntamiento ante el devenir del proyecto propio.

La fase de diseño otorga una visión gráfica del portal web que tienen en mente los responsables del proyecto.

Finalmente, la implementación adquiere un protagonismo, aunque mínimo, el cual cabe remarcar y definir.

Siguiendo las estaciones definidas, durante el transcurso del camino se han cumplido objetivos, superado contratiempos y obtenido un producto el cual se desgana con todo detalle en este documento.

Liferay se postula como herramienta sobre la que centrar el desarrollo, así como, la planificación para trabajar con Liferay. Este aporta una funcionalidad básica que abarca una cantidad de requisitos del portal facilitando así las tareas de desarrollo. Con un buen análisis previo, la fase de desarrollo se puede conseguir en un tiempo a tener en cuenta debido al potencial ofrecido por la herramienta.

Se ha intentado dejar todos los cabos atados y bien definidos para que este documento sirva como guía, tanto para el ayuntamiento, en su afán de externalizar la implementación del portal, como para los futuros desarrolladores. Pretende plasmar de forma coherente las ideas recabadas durante las distintas entrevistas y reuniones con los implicados en el portal. Gracias a esto, se espera acortar el tiempo invertido por la empresa que adquiera la adjudicación del proyecto abaratando el coste total del proyecto.

Si bien no es palpable, a los beneficios obtenidos por parte del alumno en experiencia laboral, hay que sumarle los beneficios que obtienen los implicados en la definición del portal. Consiguen esclarecer sus ideas gracias a las distintas revisiones realizadas, y con ello, entienden mejor el marco en que transcurre el desarrollo de un proyecto de las características que se citan. Así como, son conscientes de las características y funcionalidades sobre las que pueden exigir más

precisión, y sobre las que es una temeridad adentrarse sin sufrir una penalización económica considerable.

Palabras clave

Participación Ciudadana
Transparencia política
Portal municipal de Vinaros
Java 2 Enterprise Edition (J2EE)
Patrones de diseño
Framework Spring MVC
Framework Hibernate

Keywords

Citizen participation
Political transparency
Municipal portal Vinaros
Java 2 Enterprise Edition (J2EE)
Design patterns
Spring MVC Framework
Hibernate Framework

Índice general

1. Introducción	11
1.1. Contexto del proyecto	11
1.1.1. Lugar de trabajo	11
1.2. Motivación del proyecto	12
1.3. Objetivos del proyecto	13
2. Planificación del proyecto	17
2.1. Planificación temporal de las tareas	17
2.1.1. Inicial	17
2.1.2. Real	19
2.2. Estimación de recursos del proyecto	22
2.2.1. Coste RRHH	22
2.2.2. Coste Software	25
2.2.3. Coste Hardware	26
3. Análisis de frameworks, patrones de diseño J2EE y metodología	27
3.1. Metodología	27
3.1.1. ATDD (Acceptance Test Driven Development)	28
3.2. Patrones de diseño J2EE	29

3.2.1.	Introducción	29
3.2.2.	Patrones	32
3.3.	Modelo-Vista-Controldor (MVC)	37
3.3.1.	Descripción del MVC	37
3.3.2.	Funcionamiento del MVC en J2EE	38
3.4.	Frameworks J2EE	39
3.4.1.	Framework de la capa de presentación	40
3.4.2.	Framework de persistencia	46
4.	Arquitectura Liferay	49
4.1.	Introducción Portal Liferay	49
4.2.	Arquitectura de los componentes	50
4.3.	Entorno	54
5.	Análisis funcional	57
5.1.	Documento base	59
5.2.	Requisitos funcionales	66
5.3.	Historias de usuario	67
5.4.	Casos de uso	70
5.5.	Pruebas de aceptación	76
5.6.	Requisitos de datos	77
6.	Diseño	81
6.1.	Diseño arquitectónico	81
6.1.1.	Entorno de producción	81
6.1.2.	Entorno de desarrollo	82

6.1.3.	MVC del portlet a desarrollar	82
6.1.4.	Diseño de la interfaz de usuario	83
7.	Implementación	87
7.1.	Portlet desarrollado - 'Opiniones'	87
7.2.	Portlets integrados en Liferay	89
7.2.1.	Buscador	90
7.2.2.	Idiomas	90
7.2.3.	Calendar	90
7.2.4.	Foro	90
7.2.5.	Encuestas	90
7.2.6.	Formulario de contacto	91
7.3.	Base de datos	91
7.3.1.	Ejemplo de la interfaz gráfica implementada	94
8.	Conclusiones	97

Índice de figuras

2.1. Planificación inicial de tareas	18
2.2. Tareas realizadas finalmente.	21
3.1. Obtención pruebas de aceptación.	27
3.2. Proceso ATDD	28
3.3. Catálogo de patrones J2EE	31
3.4. Patrón composite view	35
3.5. Principios Spring	41
3.6. Módulos que integran el framework Spring	42
3.7. Funcionalidad básica en una aplicación Spring MVC	43
3.8. Categorías del análisis de frameworks	45
3.9. Porcentajes informe frameworks	45
3.10. Arquitectura de la relación entre persistencia y base de datos	46
3.11. Roles de las interfaces Hibernate entre capas de persistencia y lógica de negocio	48
4.1. Arquitectura lógica de Liferay	50
4.2. Arquitectura de los componentes que dispone Liferay	51
4.3. Flujo básico entre las páginas JSP y las fases principales	53
4.4. Ciclo de vida de un portlet.	53
4.5. Entorno Liferay.	54

4.6. Cuadrante de Gartner	56
5.1. Pasos para obtener pruebas de aceptación	57
5.2. Fase 1. Documento base. Visión general del proyecto.	59
5.3. Fase 2. Requisitos de alto nivel.	66
5.4. Fase 3. Historias de usuario.	67
5.5. Fase 4. Casos de uso.	70
5.6. Fase 5. Pruebas de aceptación.	76
6.1. Interacción entre tecnologías.	82
6.2. Propuesta gráfica 1.	84
6.3. Propuesta gráfica 2.	84
6.4. Parte común de todo el portal.	85
7.1. Librerías frameworks Spring y Hibernate.	92
7.2. Propiedades de configuración.	93
7.3. Interfaz gráfica a 1280px.	94
7.4. Interfaz gráfica a 800px.	94
7.5. Interfaz para tabletas.	95
7.6. Interfaz para móvil.	96

Capítulo 1

Introducción

Este apartado sirve para conocer los detalles del proyecto y las causas que han hecho posible su existencia. Se analizan sus características principales, como el contexto, la motivación y los objetivos que se esperan obtener.

1.1. Contexto del proyecto

El actual plan de estudios establece en algunas carreras la obligatoriedad de realizar prácticas, así como, la necesidad de elaborar un trabajo de fin de grado. Por ello, la universidad, junto con el alumno, se han coordinado para encontrar un lugar donde realizar la estancia. Por proximidad al lugar de residencia del alumno, se ha solicitado al ayuntamiento de la ciudad de Vinaros la realización de la estancia de prácticas.

Se puede definir formalmente un ayuntamiento como 'la corporación formada por el alcalde o intendente y los concejales que se encargan de la administración política de un municipio. El término se utiliza como sinónimo de municipalidad o corporación municipal'.

Extrapolando su significado en un marco más metafórico, el ayuntamiento es la empresa del pueblo que trabaja para el pueblo. Razón que le lleva a ayudar en todo lo posible a los ciudadanos. Al recibir la petición, el ayuntamiento aceptó de buena forma, y coordinado con la universidad, definieron un proyecto. Proyecto que permite al estudiante acercarse a una visión más próxima del mundo laboral donde deberá desenvolverse en un futuro próximo cuando termine los estudios.

1.1.1. Lugar de trabajo

Dentro de la corporación, 25 departamentos son los que van a interactuar con el departamento de informática para la realización del proyecto. De cada uno de ellos, el responsable es el encargado de participar en las reuniones y transmitir las ideas generales.

El alumno, convive con el departamento de informática. Se le asigna un escritorio, junto con su ordenador, para que la estancia sea lo más cómoda posible.

El departamento de informática se compone por 3 integrantes. Uno es el responsable, encargado de las tareas más administrativas, de gestión y de supervisar las tareas del alumno. Los otros integrantes se encargan de administrar y gestionar el entramado de ordenadores con los que los trabajadores realizan las tareas diarias. Estos dos informáticos comparten, junto al alumno, una de las dos salas asignadas para el departamento. La otra, es donde reside el responsable, y también, el lugar donde se realizan las entrevistas y reuniones.

1.2. Motivación del proyecto

Recientemente, y con motivo de elecciones, la dirección del ayuntamiento ha sufrido un cambio en el equipo de gobierno. Uno de los puntos del programa electoral era el de modernizar el portal web con el fin de ofrecer un portal acorde a las tecnologías actuales, capaz de adaptarse a cualquier dispositivo desde el cual se acceda. Además, de dar más participación a la ciudadanía.

Por ello, y de la mano del concejal responsable del área informática, el departamento de informática se ha movilizado para realizar las gestiones oportunas para la consecución de este objetivo general. De este modo se espera fomentar la participación ciudadana mediante la oferta de distintas herramientas destinadas a ello, como serían encuestas, foro, formularios para enviar consultas, etc.

En referencia a la anterior definición más informal acerca de lo que representa un ayuntamiento, donde se decía que era la empresa del pueblo y para el pueblo, la nueva política va encaminada en hacer real la definición. Se espera otorgar al ciudadano más capacidad de decisión en los distintos conflictos, aportar ideas nuevas, controlar los recursos, así como, estar informado en todo momento de los movimientos, tanto humanos como de capital, creando una especie de red social de comunicación con el ayuntamiento.

Por otro lado, el ayuntamiento dispone actualmente de 6 webs independientes y con proyección de implementar un par más. Con esta política, solo se consigue dispersar esfuerzos y distorsionar la imagen corporativa. Tras plantearse distintas opciones, el departamento de informática, conocedor de la estructura y el funcionamiento interno, propuso centralizar todos los portales web propiedad del ayuntamiento en uno solo. Con ello esperan mantener una estructura jerárquica definida, facilitar las labores de mantenimiento, abaratar costes y respetar la imagen corporativa ya definida anteriormente en otro proyecto. A grandes rasgos, en todo momento, será visible una parte común encargada de representar al ayuntamiento facilitando la navegación, y el resto de contenido será gestionado independientemente por cada responsable. Gracias a ofrecer este grado de independencia, se consiguen simular las características ofrecidas al tener portales dispersados unido a las ventajas de centralizar esfuerzos.

Las características citadas anteriormente definen la motivación institucional para llevar a cabo el proyecto.

La motivación para el proyecto que ha de realizar el alumno nace del conjunto anterior y de la necesidad de facilitar la labor al departamento de informática. Este, en su afán de externalizar el desarrollo, necesita definir las cláusulas del pliego de condiciones para poder realizar la concesión a las empresas interesadas. Por ello, es necesario llegar a un amplio nivel de análisis y diseño. El alumno ha de coordinar los distintos departamentos, gestionar y plasmar sus deseos, así como, generar un documento que transmita las ideas a los interesados. Motivo por el cual, se va se

va a hacer hincapié en analizar todos los detalles del proyecto hasta obtener un alto nivel de precisión. La implementación, dado que el ayuntamiento espera externalizar la misma, queda relegada a un segundo plano y el alumno solo implementará una pequeña funcionalidad donde pueda demostrar sus conocimientos, aunque va a dedicar mayores esfuerzos en plasmar las ideas de los implicados para la realización del futuro portal municipal.

1.3. Objetivos del proyecto

Se espera ofrecer al alumno un entorno de trabajo y unas exigencias acordes al mundo laboral. Con ello, se pretende que el alumno cumpla unos objetivos de producción y muestre rendimiento en su tarea. Gracias a esto, el alumno, afianza los conocimientos obtenidos durante su formación, a la par que, obtiene nuevos conocimientos fruto de la experiencia e investigación necesaria para superar contratiempos.

El ayuntamiento, espera obtener un documento que plasme las ideas de los responsables implicados en la creación del portal web, para así, poder negociar de forma más precisa a la hora de externalizar el proyecto.

Un proyecto, con el cual los actuales dirigentes de la corporación, pretenden situar el ayuntamiento de Vinaros entre las 100 primeras instituciones del ranking del Índice de Transparencia Municipal (ITA) de Transparency International España. Un ranking que se viene elaborando desde el año 2008 por la organización Transparencia Internacional (TI). Organización no gubernamental a escala universal dedicada a combatir la corrupción.

Van a dar especial importancia a los cuidados, tanto en lo que es transparencia de la corporación, como de contratación, en el urbanismo, y en la gestión de los recursos públicos. El comienzo es precavido, aunque esperan poder cumplir con todos los indicadores en un margen de dos años para ser lo más transparentes posible.

Por otro lado, la corporación, desea fomentar y afianzar la imagen corporativa. Una imagen ya creada y definida anteriormente. Se pretende facilitar la distinción de elementos municipales mediante la utilización de la marca propia. En la actualidad, disponen de distintos portales que son cargo del ayuntamiento, aunque, cada cual tiene una imagen y estructura. Esto propicia la distorsión de la imagen corporativa, deteriorando la misma. Sumando un mayor coste de mantenimiento. Estas razones, hacen que sea urgente marcar un plan de futuro. Plan que pasa por centralizar todos los portales creados, así como, los de nueva creación, creando una estructura jerárquica. Esta estructura jerárquica obliga a cumplir una serie de pautas y normas a todos los descendientes. Si bien habrá limitaciones, estas van a ser mínimas. Solo se restringirá la parte superior del portal para que todos tengan la misma, el restos del portal será de libre manipulación.

El portal ha de ser responsive. El diseño web responsive o adaptativo es una técnica de diseño web que busca la correcta visualización de una misma página en distintos dispositivos. Desde ordenadores de escritorio a tablets y móviles. Así, no se discrimina ningún usuario, más viendo las tendencias actuales a usar más dispositivos móviles.

El portal, al agrupar todos los departamentos y trabajadores de la corporación, así como, entidades externas, presenta un organigrama extenso. Ello implica, muchas personas interac-

tuando con el sistema donde reside información sensible, lo cual obliga a elaborar un entramado de usuarios y roles para mantener el portal seguro. La idea consiste en crear dos redes de usuarios.

Por un lado, los usuarios de la corporación, los cuales tienen credenciales en el sistema de directorio de windows Active Directory (AD), deberán poder acceder al portal con sus credenciales de la corporación. De alguna forma, el portal deberá comunicarse mediante LDAP (Lightweight Directory Access Protocol) con el sistema de directorio AD y permitir el acceso al sistema a los responsable.

Por otro lado, están los usuarios externos a la corporación. Estos se han de crear sus credenciales mediante un sistema ofrecido por el portal.

Con estas dos redes de usuarios, grupos y roles, el portal ofrecerá seguridad sobre la información sensible, permitirá gestionar y limitar el acceso a los distintos contenidos.

Continuado con los trabajadores y departamentos, que la gestión del volumen de información que mueve el ayuntamiento sea cargo del departamento de informática es inviable. En el portal actual, el departamento es el cuello de botella en lo que se refiere a actualizar el portal. Muchas de las tareas de mantenimiento y actualización son cargo de este, con lo cual, se sufren retrasos y supone una carga de trabajo para un departamento con otras funciones. Motivación que obliga a pensar en alternativas. La alternativa más aceptada por los responsables es la de repartir responsabilidades. Cada departamento cuida su parcela, así, se favorece la oferta de información y la velocidad de actualización. Con lo cual, el sistema ha de ser gestionable por los distintos usuarios internos, limitando las parcelas mediante los roles y grupos mencionados anteriormente.

Con los usuarios y la distinta información tenemos que tener en cuenta que la legislación española menciona que si recoges datos personales de tus usuarios o clientes, estás obligado a cumplir la LOPD o Ley Orgánica de Protección de Datos.

La LOPD dice que, en el caso de que guardes datos personales, estás obligado a:

- Informar a tus usuarios de qué datos personales estás recogiendo, cómo vas a utilizarlos y obtener su consentimiento para hacerlo.
- Implementar las medidas de seguridad oportunas para proteger los datos. El nivel básico de seguridad, nos obliga a:
 - Garantizar que no dejaremos que cualquier persona acceda a los datos, mediante una contraseña
 - Hacer un copia de seguridad de los datos con carácter semanal
 - Cambiar la contraseña de acceso al menos una vez al año
 - Escribir un documento que explique nuestra 'política de seguridad' (quién tiene permisos de acceso, qué base de datos se utiliza, etc?)
- Registrar el fichero de datos en la AEPD, Agencia Española de Protección de Datos

Al querer llegar a todos los usuarios interesados, el portal ha de ser accesible. La accesibilidad Web significa que personas con algún tipo de discapacidad van a poder hacer uso de la Web. En concreto, al hablar de accesibilidad Web se está haciendo referencia a un diseño Web que va a permitir que estas personas puedan percibir, entender, navegar e interactuar con la

Web, aportando a su vez contenidos. La accesibilidad Web también beneficia a otras personas, incluyendo personas de edad avanzada que han visto mermadas sus habilidades a consecuencia de la edad.

La accesibilidad Web engloba muchos tipos de discapacidades, incluyendo problemas visuales, auditivos, físicos, cognitivos, neurológicos y del habla. Gracias a el Consorcio World Wide Web (W3C), se espera cumplir con todas las pautas para tener un portal 100% accesible. El Consorcio World Wide Web (W3C) es una comunidad internacional que tiene como objetivo guiar la Web hacia su máximo potencial a través del desarrollo de protocolos y pautas que aseguren el crecimiento futuro de la Web. Debajo tratamos importantes aspectos de este objetivo, los cuales promueven la visión del W3C de Web Única.

En temas de participación ciudadana, ofrecer el mayor número de herramientas de interacción posible es un objetivo fundamental. Las distintas vías para recabar información del usuario han de estar disponibles. Desde un foro, encuestas, formularios de contactos, procesos de participación, etc, y todo ellos gestionable mediante los permisos y roles para que cada departamento pueda ser partícipe en su medida.

Finalmente, se marca el objetivo de usar software libre. La corporación lleva tiempo invirtiendo esfuerzos en la migración hacia software libre. Por ello, el nuevo proyecto está encaminado en esa dirección. Actualmente disponen de un portal realizado con la herramienta Liferay, que ha dado un rendimiento más que aceptable.

Liferay, al ser un portal de gestión de contenidos libre y de gran aceptación, ofreciendo las herramientas necesarias para la mayoría de funcionalidades propuestas, se postula como el candidato ideal para el proyecto. En mayor parte, secundado por el departamento de informática, que lo ve como una herramienta indispensable, y apoyado por los responsables por ser software libre. Por ello el objetivo principal es tener todos los portales gestionados por Liferay.

Liferay es multisite, lo cual cumple el requisito. Responsive desde su instalación, gestión de contenidos, roles de usuarios y grupos, integración de LDAP, herramientas como encuestas, un foro y otras características, hacen de este la elección para el portal a desarrollar.

Capítulo 2

Planificación del proyecto

2.1. Planificación temporal de las tareas

2.1.1. Inicial

En la figura 2.1 podemos observar las distintas tareas, así como, su duración. A grandes rasgos, el proyecto seguirá el siguiente hilo:

La primera semana (35 horas), se dedica a desarrollar la propuesta técnica. Para ello, se definirá formalmente el proyecto, se presentará el entorno donde el alumno realizará la estancia y a sus compañeros, se investigará el ámbito del proyecto para conocer sus características, y así, poder definir las tareas necesarias para desarrollar el proyecto, y finalmente, se estimarán las tareas de forma coherente. Dado que la experiencia del alumno es escasa, lo más probable es que esta planificación no se acerque a la realidad.

Una vez conocido el proyecto, el entorno y los objetivos esperados, nos sumergimos en una batería de reuniones y entrevistas para conseguir plasmar los detalles del proyecto. Se necesita encajar todos los engranajes del proyecto puesto que esta es la parte en la que más interés deposita el ayuntamiento. De esta documentación van a realizar un pliego de condiciones y negociar con las empresas a la hora de externalizar el desarrollo.

Una vez definido el proyecto, todos los participantes y sus características, se entra en una fase un poco incierta. Se espera implementar una pequeña funcionalidad para que el alumno demuestre las habilidades y conocimientos obtenidos. Dado que el ayuntamiento no hará uso posterior de la implementación, se le resta importancia y es el alumno quien decide la funcionalidad a desarrollar. A ser posible, centrada en el ámbito de participación y las nuevas funcionalidades para que los responsables puedan ver cómo quedarían sus peticiones.

Al finalizar la estancia, es momento de documentar el trabajo realizado mediante una memoria que se entregará a la universidad. Se dedicarán 120 horas a los preparativos de la memoria y la posterior presentación oral.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	Desarrollar propuesta técnica	35 horas	lun 03/08/15	vie 07/08/15	
2	Inicio	8 horas	lun 03/08/15	mar 04/08/15	
3	Definir el proyecto con el tutor y el supervisor	1 hora	lun 03/08/15	lun 03/08/15	
4	Definir la metodología de trabajo y documentación	4 horas	lun 03/08/15	lun 03/08/15	3
5	Definir estructura corporativa	3 horas	lun 03/08/15	mar 04/08/15	4
6	Documentar y planificar el proyecto	16 horas	mar 04/08/15	jue 06/08/15	
7	Revisar el contexto y buscar información	8 horas	mar 04/08/15	mié 05/08/15	2
8	Reunión con el jefe de proyecto	4 horas	mié 05/08/15	mié 05/08/15	7
9	Identificar alcance y objetivos	4 horas	mié 05/08/15	jue 06/08/15	8
10	Planificar el proyecto	11 horas	jue 06/08/15	vie 07/08/15	
11	Definición de tareas y estimación de fechas	3 horas	jue 06/08/15	jue 06/08/15	6
12	Crear diagrama de Gantt	2 horas	jue 06/08/15	vie 07/08/15	11
13	Documentar la propuesta técnica	6 horas	vie 07/08/15	vie 07/08/15	12
14	Entrega de la propuesta técnica	0 horas	vie 07/08/15	vie 07/08/15	13
15	Definición del proyecto	200 horas	lun 10/08/15	jue 17/09/15	
16	Definición de requisitos	70 horas	lun 10/08/15	vie 21/08/15	
17	Entrevistas con los remponsables implicados	21 horas	lun 10/08/15	mié 12/08/15	1
18	Definir y documentar requisitos	49 horas	jue 13/08/15	vie 21/08/15	17
19	Definir usuarios	35 horas	lun 24/08/15	vie 28/08/15	
20	Crear organigrama de la empresa	28 horas	lun 24/08/15	jue 27/08/15	16
21	Definir usuarios y roles	7 horas	vie 28/08/15	vie 28/08/15	20
22	Historias de usuario	70 horas	lun 31/08/15	vie 11/09/15	
23	Definir historias de usuario	56 horas	lun 31/08/15	mié 09/09/15	19
24	Definir escenarios	14 horas	jue 10/09/15	vie 11/09/15	23
25	Refinar	25 horas	lun 14/09/15	jue 17/09/15	
26	Refinar los requisitos y las historias de usuario	25 horas	lun 14/09/15	jue 17/09/15	22
27	Implementación	65 horas	jue 17/09/15	mié 30/09/15	
28	Preparar entorno de desarrollo	5 horas	jue 17/09/15	vie 18/09/15	15
29	Programación	50 horas	vie 18/09/15	mar 29/09/15	28
30	Pruebas	10 horas	mar 29/09/15	mié 30/09/15	29
31	Entrega	0 horas	mié 30/09/15	mié 30/09/15	30
32	Documentación y presentación del TFG	120 horas	lun 05/10/15	mié 28/10/15	
33	Redacción de informes quincenales	6 horas	lun 05/10/15	lun 05/10/15	28
34	Redacción de la memoria técnica	90 horas	lun 05/10/15	jue 22/10/15	33
35	Entrega de la memoria técnica	0 horas	jue 22/10/15	jue 22/10/15	34
36	Preparación de la presentación oral	23 horas	jue 22/10/15	mar 27/10/15	35
37	Presentación oral	1 hora	mié 28/10/15	mié 28/10/15	36

Figura 2.1: Planificación inicial de tareas

2.1.2. Real

Puedo afirmar que he vivido el refrán 'Del dicho al hecho hay un trecho'. Debido a la inexperiencia personal, a la multitud de personas que tienen voz en el proyecto, y a distintas causas, el proyecto ha seguido más bien poco la planificación inicial. Esto ha propiciado que las tareas iniciales de análisis y diseño se hayan alargado en el tiempo, restando tiempo a la implementación. Por parte del ayuntamiento no ha habido inconveniente en poner más esfuerzo en las fases iniciales debido a que la implementación, prevista de externalización, no tiene mucha importancia.

Como observamos en la figura 2.2 que muestra el trabajo realizado, y en comparación con la planificación inicial vista en la figura 2.1 se aprecian diferencias considerables.

La primera semana todo fue según lo previsto. Se realizaron las presentaciones oportunas, se conoció el ambiente de trabajo y el organigrama. Aunque la mitad del personal estaba de vacaciones, se entendió más o menos la envergadura de la corporación. Tras esta semana inicial, se obtuvo la propuesta técnica y una imagen inicial del proyecto y sus pretensiones.

El siguiente paso fue sentarnos para definir la visión más amplia del proyecto. En las reuniones participaron los responsables políticos del proyecto, el responsable del departamento de informática y el secretario de la corporación. En ellas, se decidió crear una estructura que simula un 'mapa web', para organizar el contenido de forma clara y concisa. Gracias a esto, era más fácil para todos los participantes comprender que se necesitaba. Fueron necesarias investigaciones en lo que se refiere a portales de similares características. Tras dos semanas de reuniones e investigaciones, teníamos la estructura del portal, su organización en secciones y el contenido que se ofrecería. Esta estructura la podemos encontrar detallada más adelante.

Las siguiente 17 horas se destinaron a conocer los diferentes roles y usuarios que existirán en el sistema. Se empezó de forma ambiciosa, intentando hablar con todos los departamentos y sus integrantes, definiendo a la par, los usuarios externos y sus características. Aunque, esta tarea sufrió un par de contratiempos que obligó a definir, por ahora, grupos de usuarios genéricos. En un futuro el departamento de informática realizará una investigación más a fondo según sus necesidades.

Por un lado, de los usuarios externos, no se tenía clara la forma de registro posible. En principio se deseaba distinguir entre usuarios estrictamente del municipio y los que no. Pero para ello, era necesario realizar una verificación. Se planteó un registro presencial, usar un certificado digital, o simplemente usar el portal y enviar vía correo la documentación para verificar su identidad. Surgieron dudas desde el principio, así que finalmente se desestimó realizar cualquier distinción y tener solo usuarios registrados mediante un formulario.

Por otro lado, algunos responsables de departamento estaban de vacaciones, con lo cual, no se podían detallar sus pretensiones.

Finalmente se definieron cuatro grupos de usuario:

- Administradores: Control total sobre el portal
- Usuarios internos de la corporación: Tienen control sobre la parcela del portal correspondiente a su departamento y funcionalidades legadas por el administrador

- Usuarios registrados: Pueden navegar por el portal, salvar sus preferencias de configuración y usar las herramientas de participación
- Usuarios anónimos: Sólo pueden navegar por el contenido público sin poder usar las herramientas que requieran estar logueado

Con todo definido, se dio paso a documentar la información obtenida. Primeramente se definieron los requisitos del sistema. De ellos se obtuvieron las historias de usuarios, y de estas, los casos de uso. En los dos primeros puntos se dedicó especial atención debido a los actores que había que interconectar. Aunque, cohesionar todos estos elementos y conseguir un resultado coherente nos robo más tiempo del previsto.

Tras una reunión con el supervisor, y estudiar detalladamente la situación, se decidió llegar hasta este nivel de precisión en el análisis y continuar con las fases restantes.

Con ello, se dio paso al diseño. En esta parte, se crearon un par de propuestas de la interfaz gráfica, así como, pautas generales a seguir en las diferentes secciones del portal. Las propuestas y el resultado final se pueden observar en las figuras 6.3, 6.4 y 7.3 mostradas más adelante.

La última semana se dedicó a implementar una funcionalidad para demostrar conocimientos. Ante la pronta finalización de la estancia, deje un poco de lado las buenas prácticas propuestas por la metodología seleccionada y directamente desarrolle la aplicación basándose en su funcionalidad sin atender a la calidad. En esta semana se obtuvo un portal que ofrece bastantes herramientas de participación integradas en Liferay, así como, una herramienta desarrollada por mi.

Una vez más, la falta de experiencia propia ha condicionado la consecución de los objetivos. Se ha obtenido un resultado aceptable en las distintas fases del proyecto, y una dosis de experiencia valiosa para el futuro.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	Desarrollar propuesta técnica	35 horas	lun 03/08/15	vie 07/08/15	
2	Inicio	8 horas	lun 03/08/15	mar 04/08/15	
3	Definir el proyecto con el tutor y el supervisor	1 hora	lun 03/08/15	lun 03/08/15	
4	Definir la metodología de trabajo y documentación	4 horas	lun 03/08/15	lun 03/08/15	3
5	Definir estructura corporativa	3 horas	lun 03/08/15	mar 04/08/15	4
6	Documentar y planificar el proyecto	16 horas	mar 04/08/15	jue 06/08/15	
7	Revisar el contexto y buscar información	8 horas	mar 04/08/15	mié 05/08/15	2
8	Reunión con el jefe de proyecto	4 horas	mié 05/08/15	mié 05/08/15	7
9	Identificar alcance y objetivos	4 horas	mié 05/08/15	jue 06/08/15	8
10	Planificar el proyecto	11 horas	jue 06/08/15	vie 07/08/15	
11	Definición de tareas y estimación de fechas	3 horas	jue 06/08/15	jue 06/08/15	6
12	Crear diagrama de Gantt	2 horas	jue 06/08/15	vie 07/08/15	11
13	Documentar la propuesta técnica	6 horas	vie 07/08/15	vie 07/08/15	12
14	Entrega de la propuesta técnica	0 horas	vie 07/08/15	vie 07/08/15	13
15	Definición del proyecto	230 horas	lun 10/08/15	mié 23/09/15	
16	Definir estructura y contenidos	70 horas	lun 10/08/15	vie 21/08/15	
17	Reuniones con supervisor y encargados del proyecto	35 horas	lun 10/08/15	vie 14/08/15	1
18	Investigación de portales similares	20 horas	lun 17/08/15	mié 19/08/15	17
19	Reuniones con los responsables de sección	15 horas	mié 19/08/15	vie 21/08/15	18
20	Definir usuarios	17 horas	lun 24/08/15	mié 26/08/15	
21	Crear organigrama de la empresa	11 horas	lun 24/08/15	mar 25/08/15	16
22	Definir usuarios y roles	6 horas	mar 25/08/15	mié 26/08/15	21
23	Definición de requisitos	45 horas	mié 26/08/15	jue 03/09/15	
24	Entrevistas con los responsables implicados	29 horas	mié 26/08/15	mar 01/09/15	20
25	Definir y documentar requisitos	16 horas	mar 01/09/15	jue 03/09/15	24
26	Historias de usuario	23 horas	jue 03/09/15	mié 09/09/15	
27	Definir historias de usuario	23 horas	jue 03/09/15	mié 09/09/15	23
28	Definir caso de uso	16 horas	mié 09/09/15	vie 11/09/15	
29	Documentar casos de uso	16 horas	mié 09/09/15	vie 11/09/15	26
30	Definir pruebas de aceptación	5 horas	vie 11/09/15	lun 14/09/15	
31	Documentar pruebas de aceptación	5 horas	vie 11/09/15	lun 14/09/15	28
32	Definir requisitos de datos	33 horas	lun 14/09/15	vie 18/09/15	
33	Revisar portal actual	14 horas	lun 14/09/15	mié 16/09/15	30
34	Reunión con el supervisor	3 horas	mié 16/09/15	mié 16/09/15	33
35	Documentar requisitos de datos	16 horas	mié 16/09/15	vie 18/09/15	34
36	Diseño	21 horas	vie 18/09/15	mié 23/09/15	
37	Diseño propuesta interfaz gráfica	12 horas	vie 18/09/15	mar 22/09/15	32
38	Elaboración interfaz gráfica	9 horas	mar 22/09/15	mié 23/09/15	37
39	Implementación	35 horas	mié 23/09/15	mié 30/09/15	
40	Preparar entorno de desarrollo	5 horas	mié 23/09/15	jue 24/09/15	15
41	Programación	30 horas	jue 24/09/15	mié 30/09/15	40
42	Entrega	0 horas	mié 30/09/15	mié 30/09/15	41
43	Documentación y presentación del TFG	120 horas	lun 05/10/15	mié 28/10/15	
44	Redacción de informes quincenales	6 horas	lun 05/10/15	lun 05/10/15	40
45	Redacción de la memoria técnica	90 horas	lun 05/10/15	jue 22/10/15	44
46	Entrega de la memoria técnica	0 horas	jue 22/10/15	jue 22/10/15	45
47	Preparación de la presentación oral	23 horas	jue 22/10/15	mar 27/10/15	46
48	Presentación oral	1 hora	mié 28/10/15	mié 28/10/15	47

Figura 2.2: Tareas realizadas finalmente.

2.2. Estimación de recursos del proyecto

2.2.1. Coste RRHH

Para la realización del proyecto ha intervenido el alumno, el supervisor y los distintos responsables de sección.

Al alumno se le presupone un salario bruto de 700 euros.

Salario bruto	
700.00	euros
+ 116.66	pagas prorrateadas
816.66	euros

Seguridad Social	
23.60 %	contingencias comunes
+ 6.70 %	tipo general por desempleo para un contrato temporal a tiempo completo
+ 0.20 %	Fogasa
+ 0.70 %	Formación profesional
31.20 %	

31.2 % de 816.66 euros = 254.80 euros

Total	
816.66	salario
+ 254.80	Seguridad social
1071.46	euros/mes

Horas mensuales:

7 horas diarias * 5 días semanales * 4 semanas = 140 horas mensuales

1 hora = 1071.46 / 140 = **7.65 euros/hora**

El supervisor tiene un salario de 3200 euros brutos

Salario bruto	
3200.00	euros
+ 533.33	pagas prorrateadas
3733.33	euros

Seguridad Social	
23.60 %	contingencias comunes
+ 5.50 %	tipo general por desempleo para un contrato indefinido
+ 0.20 %	Fogasa
+ 0.70 %	Formación profesional
30.00 %	

30 % de 3733.33 euros = 1120 euros

Total	
3733.33	salario
+ 1120	Seguridad social
4853.33	euros/mes

Horas mensuales:

8 horas diarias * 5 días semanales * 4 semanas = 160 horas mensuales

1 hora = 4853,33/160 = **30,33 euros/hora**

Los responsables tienen un salario de 2600 euros brutos

Salario bruto	
2600.00	euros
+ 433.33	pagas prorrateadas
3033.33	euros

Seguridad Social	
23.60 %	contingencias comunes
+ 5.50 %	tipo general por desempleo para un contrato indefinido
+ 0.20 %	Fogasa
+ 0.70 %	Formación profesional
30.00 %	

30 % de 3033 euros = 910 euros

Total	
3033.33	salario
+ 910	Seguridad social
3943.33	euros/mes

Horas mensuales:

8 horas diarias * 5 días semanales * 4 semanas = 160 horas mensuales

1 hora = $3943/160 = \mathbf{24.64 \text{ euros/hora}}$

Coste total para le empresa:

Alumno		Supervisor		Responsables	
300	horas	50	horas	20	horas
* 7.65	euros/hora	* 30.33	euros/hora	* 24.64	euros/hora
2295	euros	1516.50	euros	492.80	euros
$2295 + 1516.50 + 492.80 = \mathbf{4304.30 \text{ euros.}}$					

2.2.2. Coste Software

Para desarrollar el proyecto se han intentado utilizar el máximo número de software Open Source.

- **TexStudio:** es un editor de LaTeX de código abierto y multiplataforma. Proporciona un soporte moderno de escritura, como la corrección ortográfica interactiva, plegado de código y resaltado de sintaxis. Se ha utilizado para la redacción de los distintos documentos oficiales para la universidad (Informes quincenales, Propuesta técnica, memoria).
- **Microsoft Project:** (o MSP) es un software de administración de proyectos diseñado, desarrollado y comercializado por Microsoft para asistir a administradores de proyectos en el desarrollo de planes, asignación de recursos a tareas, dar seguimiento al progreso, administrar presupuesto y analizar cargas de trabajo. Se utiliza con la licencia educativa entregada durante la estancia de estudios. Se ha utilizado para crear la planificación de tareas.
- **Google drive:** Se ha usado google drive para la documentación de los distintos procesos durante el desarrollo del proyecto, así como, para la realización de la presentación.
- **Liferay:** es un portal de gestión de contenidos de código abierto escrito en Java. Se creó en 2000 en principio como solución para las organizaciones sin ánimo de lucro. Sobre este gira todo el entramado del proyecto. Es la piedra angular que nos proporciona herramientas y facilita la programación, obteniendo el producto con menor esfuerzo.
- **Eclipse:** es una plataforma de desarrollo, diseñada para ser extendida de forma indefinida a través de plug-ins. Fue concebida desde sus orígenes para convertirse en una plataforma de integración de herramientas de desarrollo. No tiene en mente un lenguaje específico, sino que es un IDE genérico, aunque goza de mucha popularidad entre la comunidad de desarrolladores del lenguaje Java usando el plug-in JDT (Java development tools) que viene incluido en la distribución estándar del IDE.
Proporciona herramientas para la gestión de espacios de trabajo, escribir, desplegar, ejecutar y depurar aplicaciones. Se ha usado para desarrollar la funcionalidad específica del portal.
Se ha integrado en eclipse el plugin que ofrece Lifera y, así como, el plugin Software Development Kit (SDK) que nos permiten desarrollar para Lifera y.
- **Tomcat:** es un contenedor web con soporte de servlets y JSPs (JavaServer Pages). Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache.
Tomcat puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.
Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.
Se ha instalado un servidor Tomcat y posteriormente integrado en Eclipse para la fase de desarrollo, así como, se utiliza un servidor Tomcat en producción para servir el portal a los usuarios.

- **MySQL:** es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. Se ha utilizado durante el desarrollo como sistema gestor de base de datos. Se ha seleccionado este por simplicidad y el conocimiento sobre él. Liferay ofrece la posibilidad de migrar sus datos de forma sencilla y eficaz. En el momento de poner la aplicación en producción los datos se migrarán a la base de datos Oracle de la corporación.
- **Oracle:** La corporación dispone en la actualidad de bases de datos Oracle instaladas en Oracle RAC. Oracle RAC permite que múltiples computadoras ejecuten el software de SGBD de Oracle simultáneamente mientras acceden a una base de datos individual. Esto se llama una base de datos en grupo (cluster o clustered). Por ello, en producción se ha aprovechado la base de datos corporativa y se migran los datos.

La inversión en herramientas para este proyecto ha sido nula. Se ha desarrollado aprovechando los recursos de los que disponía el ayuntamiento y de herramientas Open Source gratuitas.

2.2.3. Coste Hardware

Para el desarrollo, el ayuntamiento ha puesto al servicio del alumno un PC de sobremesa que tenían disponible en el almacén.

Es un HP junto a un monitor de 20”:

- Procesador Intel® Core i3
- Con monitor de 20”
- Memoria 4 GB DDR3L SDRAM
- Disco duro SATA de 500GB 5400 rpm
- Gráficos Intel HD Graphics 4400

En cuanto a la parte de producción, el ayuntamiento dispone de un entramado en funcionamiento el cual está capacitado para satisfacer las necesidades del portal. En la actualidad, existe un portal Liferay en funcionamiento en un servidor virtual, corriendo sobre el sistema operativo Suse Enterprise Server para VMware. Este acoge el servidor Tomcat que contendrá el portal. También disponen de Oracle Rac para almacenar la base de datos Oracle. De este modo, aprovechando la instalación configurada para el portal existente, el nuevo portal correrá sin ninguna inversión extra.

Por lo tanto, el coste final del proyecto realizado supondrá al ayuntamiento la cantidad de **4304.30 euros**.

Capítulo 3

Análisis de frameworks, patrones de diseño J2EE y metodología

3.1. Metodología

La elección ha sido la metodología de desarrollo guiado por test (ATDD).

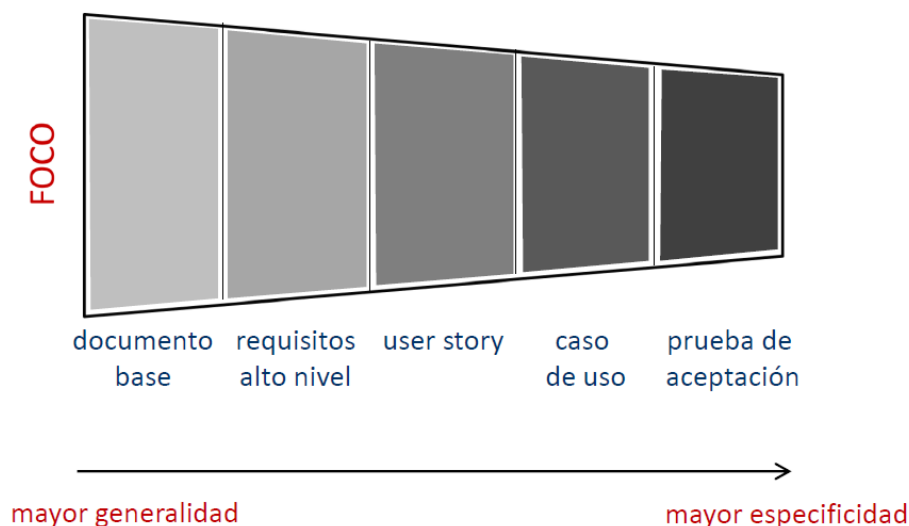


Figura 3.1: Obtención pruebas de aceptación.

El procedimiento esperado se basa en obtener las pruebas de aceptación necesarias para el desarrollo mediante ATDD que se obtuvo siguiendo el proceso que se observa en la figura 3.1, donde va de mayor generalidad a mayor especificación. Se ha conseguido seguir las pasos con mayor precisión en las fases iniciales, aunque en la parte final, no se pudo poner mucho esfuerzo

en las pruebas de aceptación.

A continuación se habla de la metodología ATDD por ser la escogida para el desarrollo. Aunque, a la hora de la verdad, y con el tiempo del proyecto consumido mayormente, no se ha seguido los consejos y buenas prácticas propuestos por la metodología. Este hecho ha servido para ver que la teoría dista de la realidad, y para futuros proyectos se va a tener más en cuenta la organización y planificación, así como, las buenas prácticas.

3.1.1. ATDD (Acceptance Test Driven Development)

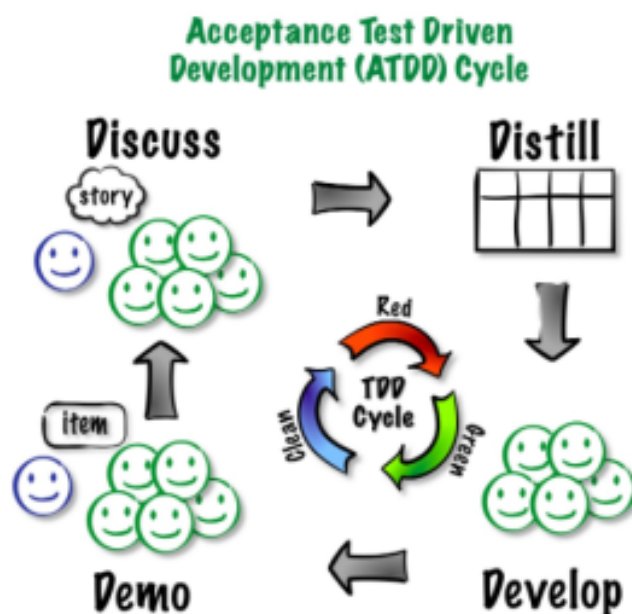


Figura 3.2: Proceso ATDD

Como se ve en la figura 3.2 y después de tener una lista de requisitos definida se ejecuta el siguiente ciclo:

Elegir un requisito: Se elige de una lista el requerimiento que se cree que nos dará mayor conocimiento del problema y que a la vez sea fácilmente implementable.

Escribir una prueba: Se comienza escribiendo una prueba para el requisito. Para ello el programador debe entender claramente las especificaciones y los requisitos de la funcionalidad que está por implementar. Este paso fuerza al programador a tomar la perspectiva de un cliente considerando el código a través de sus interfaces.

Verificar que la prueba falla: Si la prueba no falla es porque el requerimiento ya estaba implementado o porque la prueba es errónea.

Escribir la implementación: Escribir el código más sencillo que haga que la prueba funcione. Se usa la metáfora "Déjelo simple" ("Keep It Simple, Stupid" (KISS)).

Ejecutar las pruebas automatizadas: Verificar si todo el conjunto de pruebas funciona co-

rrectamente.

Eliminación de duplicación: El paso final es la refactorización, que se utilizará principalmente para eliminar código duplicado. Se hacen de a una vez un pequeño cambio y luego se corren las pruebas hasta que funcionen.

Actualización de la lista de requisitos: Se actualiza la lista de requisitos tachando el requisito implementado. Asimismo se agregan requisitos que se hayan visto como necesarios durante este ciclo y se agregan requerimientos de diseño (P. ej que una funcionalidad esté desacoplada de otra).

3.2. Patrones de diseño J2EE

En esta sección se va a hablar de patrones de diseño J2EE. Estos son importantes para construir aplicaciones J2EE, de un modo profesional, mediante técnicas avanzadas de análisis, diseño y desarrollo. Liferay, herramienta seleccionada para el portal está implementa en Java. Gracias a estos patrones conoceremos las buenas prácticas y un poco mejor el funcionamiento interno de Liferay.

3.2.1. Introducción

Un patrón es una solución de diseño no trivial que sea efectiva y reutilizable. Efectiva porque ha servido para resolver el problema de diseño anteriormente y reutilizable porque la solución es la misma para problemas similares.

Los patrones a grandes rasgos pretenden:

- Proporcionar un catálogo de elementos reutilizables en el diseño de sistemas software
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente
- Estandarizar la forma en que se realiza el diseño
- Formar un vocabulario común entre diseñadores
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando el conocimiento existente

Hablando de los patrones J2EE, están orientados específicamente a problemas comunes de las aplicaciones J2EE. Algunos están basados en los patrones originales y, por otro lado, otros son más específicos para problemas que surgen específicamente en J2EE. Sea por el tipo de aplicación a desarrollar en la plataforma o por las características de la tecnología.

La elección de una arquitectura u otra condicionarán el diseño, aunque, la utilización del patrón arquitectónico MVC (Modelo-Vista-Controlador) y los patrones de diseño J2EE, harán las aplicaciones escalables facilitando futuras ampliaciones, migraciones y cambios en general.

Clasificación según su nivel de abstracción:

- Patrón arquitectónico: aconsejan la arquitectura general que ha de seguir una aplicación, por ejemplo, el patrón Modelo-Vista-Controlador (MVC) que aconseja la arquitectura global de una aplicación interactiva.
- Patrón de diseño: explican como resolver un problema concreto de diseño, como por ejemplo, el patrón Data Access Object (DAO) que permite abstraer y encapsular el acceso a un repositorio de datos (BD relacional, BD orientada a objetos, etc).

El diseño de aplicaciones web basadas en los patrones J2EE se organizan alrededor de la utilización de diversos elementos: un controlador frontal, dispatchers, vistas compuestas, vistas (JSPs) y los helpers de las vistas (JavaBeans).

La figura 3.3 muestra una representación gráfica del catálogo de patrones principales de J2EE (Core J2EE Patterns):

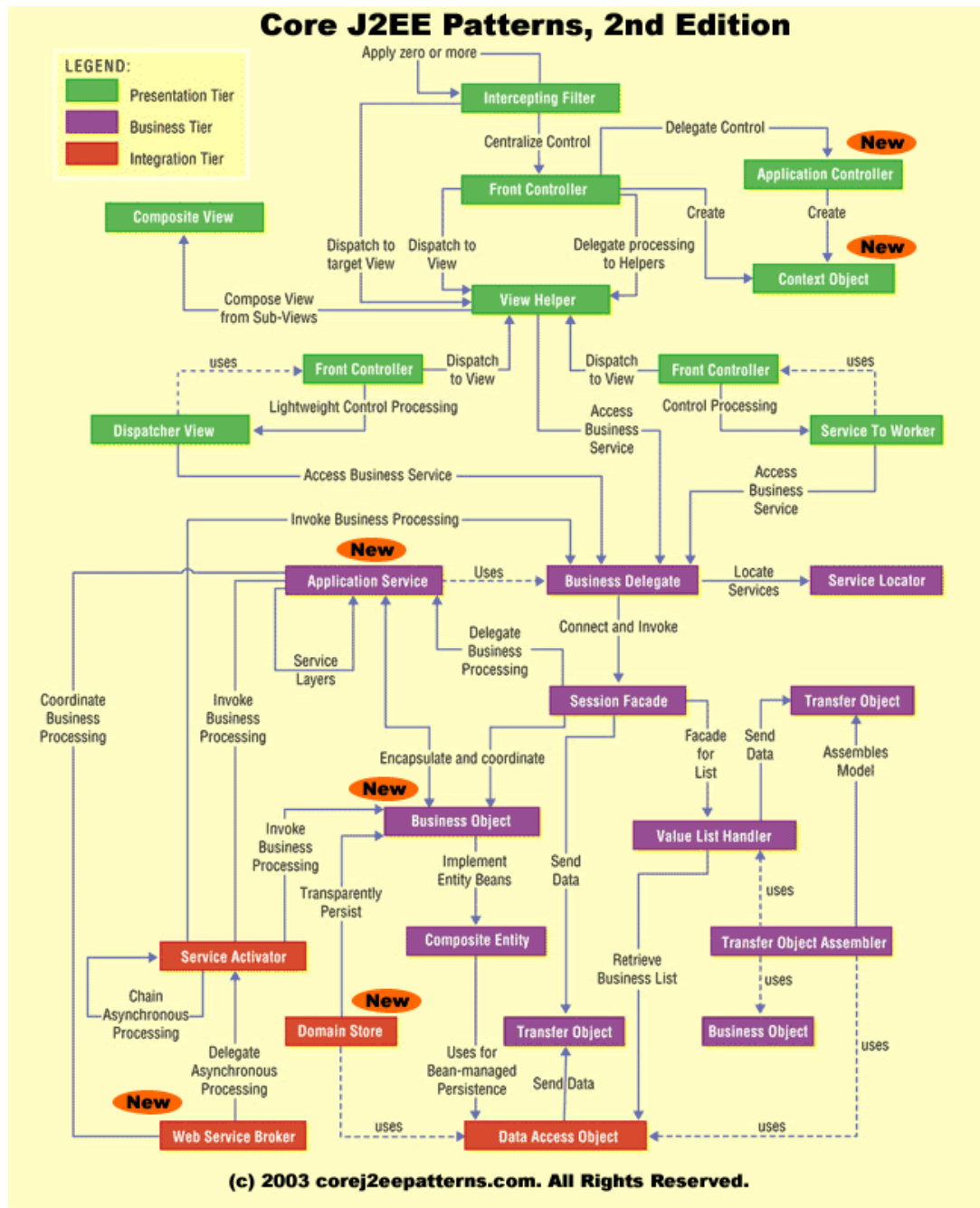


Figura 3.3: Catálogo de patrones J2EE

3.2.2. Patrones

A continuación se presentan algunos patrones dado que existen multitud de ellos que proporcionan soluciones a problemas distintos. Los patrones seleccionados facilitan las tareas comunes de comunicación y control de una aplicación web. Son utilizados internamente por Liferay, evitando que el desarrollador tenga que invertir esfuerzos en dicha parcela, pudiendo dedicarse a las tareas particulares de su aplicación. Otro, como el patrón Data Access Object (DAO) es utilizado mediante el framework de persistencia Hibernate seleccionado para el desarrollo del portlet para la aplicación.

Intercepting Filter:

Contexto

El mecanismo de manejo de peticiones de la capa de presentación recibe muchos tipos diferentes de peticiones, cada uno de los cuales requiere varios tipos de procesamiento. Algunas peticiones simplemente requieren su reenvío al componente manejador apropiado, mientras que otras peticiones deben ser modificadas, auditadas, o descomprimidas antes de su procesamiento posterior.

Solución

Crear filtros conectables para procesar servicios comunes de una forma estándar sin requerir cambios en el código principal del procesamiento de la petición. Los filtros interceptan las peticiones entrantes y las respuestas salientes, permitiendo un pre y post-procesamiento. Podemos añadir y eliminar estos filtros a discreción, sin necesitar cambios en nuestro código existente. Podemos, en efecto, decorar nuestro procesamiento principal con una variedad de servicios comunes, como la seguridad, el logging, el depurado, etc. Estos filtros son componentes independientes del código de la aplicación principal, y pueden añadirse o eliminarse de forma declarativa. Por ejemplo, se podría modificar un fichero de configuración de despliegue para configurar una cadena de filtros. Cuando un cliente pide un recurso que corresponde con este mapeo de URL configurado, se procesa cada filtro de la cadena antes de poder invocar el recurso objetivo.

Front Controller:

Contexto

El mecanismo de manejo de peticiones de la capa de presentación debe controlar y coordinar el procesamiento de todos los usuarios a través de varias peticiones. Dichos mecanismos de control se pueden manejar de una forma centralizada o descentralizada.

Solución

Usar un controlador como el punto inicial de contacto para manejar las peticiones. El controlador maneja el control de peticiones, incluyendo la invocación de los servicios de seguridad como la autenticación y autorización, delegar el procesamiento de negocio, controlar la elección de una vista apropiada, el manejo de errores, y el control de la selección de estrategias de creación de contenido.

El controlador proporciona un punto de entrada centralizado que controla y maneja las peticiones Web. Centralizando los puntos de decisión y control, el controlador también ayuda a reducir la cantidad de código Java, llamadas a scripts, embebidos en la página JavaServer Pages (JSP). Centralizar el control en el controlador y reduciendo la lógica de negocios en la vista permite

reutilizar el código entre peticiones. Es una aproximación preferible a la alternativa de embeber código en varias vistas porque esta aproximación trata con entornos más propensos a errores, y de reutilización del tipo copiar-y-pegar.

Típicamente, un controlador se coordina con un componente dispatcher. Los dispatchers son responsable del control de la vista y de la navegación. Así, un dispatcher elige la siguiente vista por el usuario y dirige el control al recurso. Los dispatchers podrían encapsularse directamente dentro del controlador o se puede extraer en un componente separado.

Aunque el patrón Front Controller sugiere la centralización del manejo de peticiones, no limita el número de manejadores en el sistema, como lo hace Singleton. Una aplicación podría utilizar varios controladores en un sistema, cada uno mapeado a un conjunto de servicios distintos.

View Helper:

Contexto

El sistema crea el contenido de la presentación, lo que requiere el procesamiento de datos de negocio dinámicos.

Solución

Una vista contiene código de formateo, delegando sus responsabilidades de procesamiento en sus clases de ayuda, implementadas como JavaBeans o etiquetas personalizadas. Las clases de ayuda o helpers también almacenan el modelo de datos intermedio de la vista y sirven como adaptadores de datos de negocio.

Hay varias estrategias para implementar el componente de la vista. La estrategia JSP View sugiere utilizar una JSP como el componente vista. Esta es la estrategia preferida, y es la que se utiliza más comúnmente. La otra estrategia principal es la estrategia Servlet View, que utiliza un servlet como la vista.

Encapsular la lógica de negocio en un helper en lugar de hacerlo en la vista hace que nuestra aplicación sea más modular y facilita la reutilización de componentes. Varios clientes, como controladores y vistas, podrían utilizar el mismo helper para recuperar y adaptar estados del modelo similares para su presentación en varias vistas. La única forma de reutilizar la lógica embebida en una vista es copiando y pegando en cualquier lugar. Además, la duplicación mediante copiar-y-pegar hace que un sistema sea difícil de mantener, ya que necesitamos corregir en muchos sitios el mismo error potencial.

Una señal de que podríamos necesitar aplicar este patrón al código existente es cuando el código scriptlet domina en la vista JSP. Ya que el objetivo general cuando se aplica este patrón, es el particionamiento de la lógica de negocio fuera de la vista. Mientras alguna lógica está mejor cuando se encapsula dentro de objetos helper, otra lógica está mejor situándose en un componente centralizado que se sitúa delante de las vistas y los helpers – esta podría la lógica que sea común entre varias peticiones, como los chequeos de autenticación o servicios de logs, por ejemplo.

Si no se emplea un controlador separado en la arquitectura, o si no se utiliza para manejar todas las peticiones, entonces el componente vista se convierte en el punto de contacto inicial para manejar algunas peticiones. Para ciertas peticiones, particularmente aquellas que implican un procesamiento mínimo, este escenario funcionará bien. Típicamente, esta situación ocurre cuando páginas que están basadas en información estática, como la primera de un conjunto de páginas que se servirán a un usuario para obtener alguna información.

El patrón View Helper se enfoca en recomendar formas de particionar las responsabilidades de nuestras aplicaciones.

Composite View:

Contexto

Las páginas Web sofisticadas presentan contenido de varias fuentes de datos, utilizando varias subvistas que completan una sola página. Además, varios individuos con diferentes habilidades contribuyen al desarrollo y mantenimiento de esas páginas Web.

Solución

Utilizar vistas compuestas que se componen de varias subvistas atómicas. Cada componente de la plantilla se podría incluir dinámicamente dentro del total y la distribución de la página se maneja independientemente del contenido.

Esta solución promueve la creación de una vista compuesta basada en la inclusión y sustitución de fragmentos de plantilla modulares tanto estáticos como dinámicos. También promueve la reutilización de porciones atómicas de la vista asegurando un diseño modular. Es apropiado utilizar este patrón para generar páginas que muestran componentes que podrían combinarse en una gran variedad de formas. Este escenario ocurre, por ejemplo, con sites de portal que incluye numerosas subvistas independientes, como noticias, información del tiempo, y valores de stocks en una sola página. La distribución de la página se maneja y modifica de forma independiente al contenido de las subvistas.

Otro beneficio de este patrón es que los diseñadores Web pueden hacer un prototipo de la distribución de la site, conectando contenido estático en todas las regiones de la plantilla. Según va progresando el desarrollo de la site, ese contenido estático se puede sustituir por el contenido dinámico.

La siguiente figura 3.4 muestra una captura de pantalla de la página inicial de Sun, java.sun.com. Se identifican cuatro regiones: Navegación, Búsqueda, Contenido y Cabeceras. Aunque el contenido de cada una de estas subvistas podría estar originado desde diferentes fuentes de datos, se unen para crear un sólo página compuesta.

Este patrón también tiene inconvenientes. Provoca una sobrecarga en el entorno de ejecución, un precio que hay que pagar por el incremento de flexibilidad que proporciona. La utilización de un mecanismo de distribución más sofisticado también trae consigo algunos problemas de manejabilidad y desarrollo, ya que hay más artefactos que mantener y un cierto nivel indirecto de implementación que entender.

Gestiona los distintos elementos de la vista mediante una plantilla que realiza la representación de las vistas más gestionables. Por esta razón se intenta utilizar vistas compuestas que están formadas por múltiples subvistas. El abuso de estas divisiones puede tener un efecto negativo en el rendimiento por lo que es conveniente tener un compromiso entre modularidad y mantenimiento frente a rendimiento.

Dispatcher View:

Contexto

El sistema controla el flujo de ejecución y accede al proceso de presentación, que es el responsable de generar el el contenido dinámico.

Nota: El patrón Dispatcher View, igual que el patrón Service to Worker, describe una combinación común de otros patrones del catálogo. Estos dos macro-patrones describen la combinación de un controlador y un dispatcher con vistas y helpers. Aunque describen esta estructura común, cada uno enfatiza un uso diferente de los patrones.

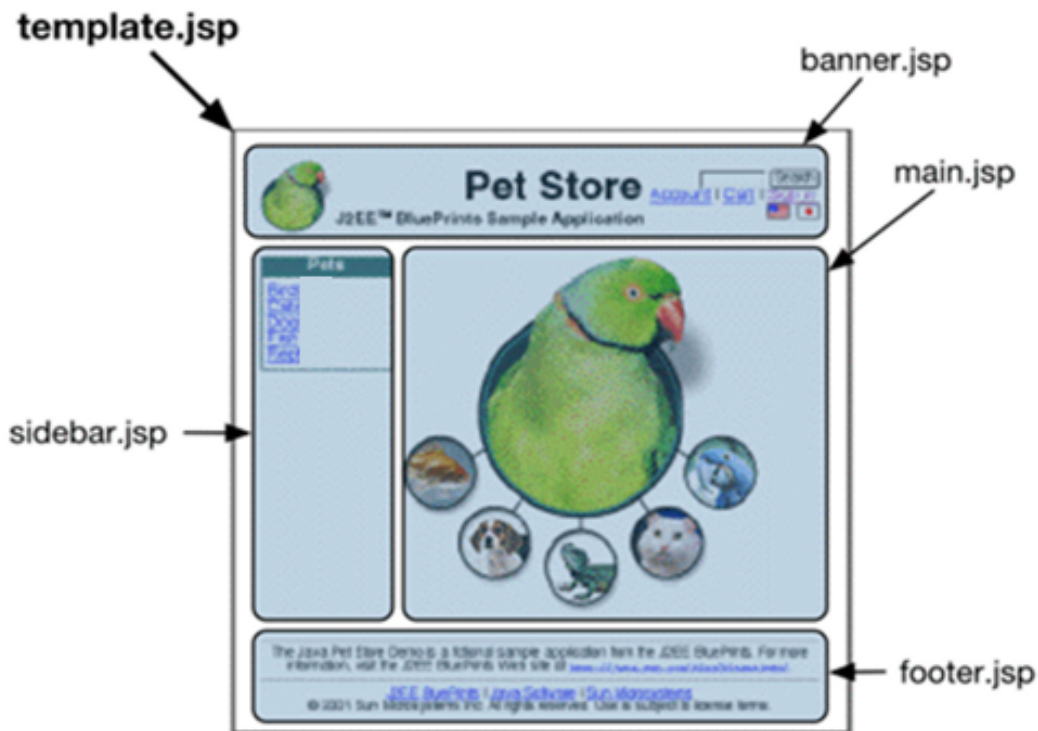


Figura 3.4: Patrón composite view

Solución

Combinar un controlador y un dispatcher con vistas y helpers para manejar peticiones de clientes y preparar una presentación dinámica como respuesta. Los controladores delegan la recuperación de contenido en los helpers, que manejan el relleno del modelo intermedio para la vista. Un dispatcher es el responsable del control de la vista y la navegación y puede encapsularse dentro de un controlador o de un componente separado.

Dispatcher View describe la combinación de los patrones Front Controller y View Helper con un componente dispatcher. Aunque este patrón y Service to Worker describen una estructura similar, ambos sugieren diferentes divisiones de la labor entre los componentes. El controlador y el dispatcher tienen responsabilidades limitadas, comparadas con el patrón Service to Worker, porque la lógica de procesamiento y de control de vista son básicas. Además, si no se considera necesario el control de los recursos subyacentes, se puede eliminar el controlador y el dispatcher se podría mover dentro de una vista.

Aunque los patrones Service to Worker y Dispatcher View representan una combinación de otros patrones del catálogo, el primero garantiza con su nombre una comunicación eficiente entre los desarrolladores. Mientras que el segundo sugiere una recuperación de contenido relegada al momento de procesamiento de la vista.

En el patrón Dispatcher View, el dispatcher normalmente juega un rol moderado en el control de la vista. En el patrón Service to Worker, el dispatcher juega un rol algo más elevado en el control de la vista.

Un rol limitado para el dispatcher ocurre cuando no se utilizan recursos exteriores para poder elegir la vista. La información encapsulada en la petición es suficiente para determinar la vista a la que despachar la petición. Por ejemplo:

http://some.server.com/servlet/Controller?next=login.jsp

La única responsabilidad del componente dispatcher en este caso es reenviar a la vista login.jsp. Un ejemplo del dispatcher jugando un rol moderado es el caso donde el cliente envía una petición directamente al controlador con un parámetro de consulta que describe una acción a realizar:

http://some.server.com/servlet/Controller?action=login

Aquí la responsabilidad del dispatcher es traducir el nombre lógico login en el nombre del recurso de una vista apropiada, como login.jsp, y reenviar a esa vista. Para conseguir esta traducción, el dispatcher podría acceder a recursos como un fichero de configuración XML que especifica las vistas apropiadas a mostrar.

Por otro lado, en el patrón Service to Worker, el dispatcher podría invocar servicios de negocio para determinar la vista apropiada que se debe mostrar. La estructura compartida de Service to Worker y Dispatcher View consiste en un controlador trabajando con un dispatcher, vistas y helpers.

Session Facade:

Contexto

Clases Java comunes o EJB encapsulan lógica y datos de negocios, exponiendo sus interfaces y la complejidad de los servicios distribuidos a la capa cliente.

Solución

Usar un session bean (para el caso de aplicaciones EJB) o una clase java común que encapsule la complejidad de las interacciones entre los objetos de negocio participantes en un flujo de trabajo (workflow). El session façade maneja los objetos de negocios y proporciona un servicio de acceso uniforme a los cliente. Es decir, el cliente (JSP, Swing, etc) no tratará con los EJB ni con la complejidad del acceso remoto, ni con JDBC, sino que trabajará con colecciones y Value's Object.

Data Access Object:

Contexto

El acceso a los datos varía dependiendo de la fuente de los datos. El acceso al almacenamiento persistente, como una base de datos, varía en gran medida dependiendo del tipo de almacenamiento (bases de datos relacionales, bases de datos orientadas a objetos, ficheros planos, etc.) y de la implementación del vendedor.

Solución

Utilizar un Data Access Object (DAO) para abstraer y encapsular todos los accesos a la fuente de datos. El DAO maneja la conexión con la fuente de datos para obtener y almacenar datos. El DAO implementa el mecanismo de acceso requerido para trabajar con la fuente de datos. Esta fuente de datos puede ser un almacenamiento persistente como una RDBMS, un servicio externo como un intercambio B2B, un repositorio LDAP, o un servicio de negocios al que se accede mediante CORBA Internet Inter-ORB Protocol (IIOP) o sockets de bajo nivel. Los componentes de negocio que tratan con el DAO utilizan un interfaz simple expuesto por el DAO para sus clientes. El DAO oculta completamente los detalles de implementación de la fuente de datos a sus clientes. Como el interface expuesto por el DAO no cambia cuando cambia la implementación de la fuente de datos subyacente, este patrón permite al DAO adaptarse a diferentes esquemas de almacenamiento sin que esto afecte a sus clientes o componentes de

negocio. Esencialmente, el DAO actúa como un adaptador entre el componente y la fuente de datos.

3.3. Modelo-Vista-Controldor (MVC)

El patrón de arquitectura MVC (Modelo Vista Controlador) es un patrón que define la organización independiente del Modelo (Objetos de Negocio), la Vista (interfaz con el usuario u otro sistema) y el Controlador (controlador del workflow de la aplicación).

Se ha utilizado este patrón mediante el framework Spring MVC en el desarrollo del plugin portlet que desarrolla una funcionalidad propia para Liferay. En la sección de implementación se entra en más detalles.

3.3.1. Descripción del MVC

El patrón de arquitectura 'modelo vista controlador', es una filosofía de diseño de aplicaciones, compuesta por:

Modelo:

- Contiene el núcleo de la funcionalidad (dominio) de la aplicación
- Encapsula el estado de la aplicación
- No sabe nada / independiente del Controlador y la Vista

Vista:

- Es la presentación del Modelo
- Puede acceder al Modelo pero nunca cambiar su estado
- Puede ser notificada cuando hay un cambio de estado en el Modelo

Controlador:

- Reacciona a la petición del Cliente, ejecutando la acción adecuada y creando el modelo pertinente

Para entender cómo funciona nuestro patrón Modelo-Vista-Controlador, se debe entender la división a través del conjunto de estos tres elementos y cómo estos componentes se comunican unos con los otros y con otras vistas y controladores externos al modelo principal. Para ello, es

importante saber que el controlador interpreta las entradas del usuario (tanto teclado como el ratón), enviado el mensaje de acción al modelo y a la vista para que se proceda con los cambios que se consideren adecuados

Comunicación

El modelo, la vista y el controlador deben comunicarse de una manera estable los unos con los otros, de forma que sea coherente con las iteraciones que el usuario realizará. Como es lógico la comunicación entre la vista y el controlador es bastante básica pues están diseñados para operar juntos, pero los modelos se comunican de una manera diferente, un poco más sutil

Modelo pasivo

No es necesario para el modelo tener alguna disposición a él, simplemente basta con tener en cuenta su existencia. El modelo no tiene ninguna responsabilidad para comunicar los cambios a la vista porque ocurren solo por orden del usuario, por lo que esta función la llevará a cabo el controlador porque será el que interprete las órdenes de este usuario debido a que solo debe comunicar que algo ha cambiado. Por esto, el modelo es se encuentra en modo inconsciente y su participación en este caso es irrisoria.

Unión del modelo con la vista y el controlador

Como no todos los modelos pueden ser pasivos, se necesita algo que comunique al controlador y a la vista, por lo que en este caso, sí que se necesita el modelo, ya que solo este puede llevar a cabo los cambios necesarios al estado actual en el que estos se encuentran.

Al contrario que el modelo, que puede ser asociado a múltiples asociaciones con otras vistas y controladores, cada vista solo puede ser asociada a un único controlador, por lo que han de tener una variable de tipo controller que notificara a la vista cual es su controlador o modelo asignado. De igual manera, el controlador tiene una variable llamada View que apunta a la vista. De esta manera, pueden enviarse mensajes directos el uno al otro y al mismo tiempo, a su modelo.

Al final, la vista es quien lleva la responsabilidad de establecer la comunicación entre los elementos de nuestro patrón MVC. Cuando la vista recibe un mensaje que concierne al modelo o al controlador, lo deja registrado como el modelo con el cual se comunicara y apunta con la variable controller al controlador asignado, enviándole al mismo su identificación para que el controlador establezca en su variable view el identificador de la vista y así puedan operar conjuntamente. El responsable de deshacer estas conexiones, seguirá siendo la vista, quitándose a sí misma como dependiente del modelo y liberando al controlador.

3.3.2. Funcionamiento del MVC en J2EE

Podríamos diferenciar el funcionamiento del patrón MVC dentro de J2EE en tres puntos:

Captura de la petición en el controlador

La aplicación recibe peticiones que son centralizadas en el Controlador. Este es el encargado de interpretar, a partir de la URL de la solicitud, el tipo de operación a realizar. Normalmente, esto se hace analizando el valor de algunos parámetros que se envían junto a la URL de la

petición y que se utiliza con esta finalidad.

Procesamiento de la petición

Una vez determinada la operación por el Controlador, procede a ejecutar las acciones pertinentes invocando los distintos métodos propuestos por el Modelo. Dependiendo de las acciones a realizar, el Modelo necesitará gestionar los datos enviados por el cliente en la petición y que son proporcionados por el Controlador. De la misma forma, los resultados generados por el Modelo será enviado al Controlador.

Generación de respuestas

Los resultados retornados por el Modelo al Controlador son guardados por este último en una variable de petición, sesión o aplicación, según el alcance que tiene. A continuación, el Controlador invoca a la página JSP que se encarga de generar la vista correspondiente, esta página accede a la variable de ámbito donde estén almacenados los resultados y los utilizará para generar dinámicamente la respuestas XHTML para enviar al cliente.

3.4. Frameworks J2EE

Un framework es un conjunto de clases e interfaces que cooperan para solucionar un tipo específico de problema software. Un framework tiene las siguientes características:

- Consta de múltiples clases o componentes, cada uno de los cuales pueden proveer una abstracción de un determinado concepto
- Define como estas abstracciones trabajan juntas para solucionar el problema
- Sus componentes son reutilizables
- Organizan patrones a alto nivel

Un buen framework deberá proveer de un comportamiento genérico el cual diferentes tipos de aplicaciones puedan hacer uso de este.

Muchos de los framework J2EE ya implementa los mecanismos de comunicación entre las capas siguiendo el MVC.

En esta sección se habla del framework usado en la capa de presentación y en la capa de persistencia. EL primero es Spring MVC, el cual facilita la organización y funcionamiento interno. Sigue el esquema propuesto por el patrón MVC (Modelo-Vista-Controlador), por lo que el desarrollo será escalable y de fácil mantenimiento.

EL segundo, es Hibernate, y este va a facilitar el trabajo de comunicación y mantenimiento de la información en la base de datos.

Gracias a ellos, se centraran esfuerzos en la lógica de negocio dejando estos elementos en sus manos.

3.4.1. Framework de la capa de presentación

Spring

Para empezar, se ha de decir que Spring no es un framework de desarrollo web. Dispone de módulos para el desarrollo web como Spring MVC y Spring Web Flow que están entre los mejores, pero por encima de esto, Spring es un framework para el desarrollo de aplicaciones empresariales Java, sea web o no. La idea es que Spring se encargue de las tareas de infraestructura para que los desarrolladores se centren en resolver el problema del dominio. Spring framework no obliga a usar un modelo de programación en particular, no obstante se ha popularizado en la comunidad de programadores Java, siendo la alternativa al modelo Enterprise Java Bean. El diseño del framework ofrece mucha libertad a los desarrolladores en Java y soluciones muy bien documentadas y fáciles de usar. Aunque las características fundamentales de estos frameworks pueden utilizar cualquier aplicación hecha en Java, existiendo muchas extensiones y mejoras para construir aplicaciones basadas en web con la plataforma empresarial J2EE.

Algunas de las principales características de Spring son:

- Flexible: se puede usar cualquier componente o módulo de Spring teniendo siempre la posibilidad de integración con otros frameworks populares sin problemas. Es decir, la herramienta no obliga a usarlo en toda su magnitud, sino, que se puede utilizar solo lo que guste.
- Débilmente acoplado: no fuerza a utilizar el framework, si no, se puede usar sus proyectos como se necesite.
- Altamente cohesivo: cada módulo se centra en lo que tiene que hacer de forma dedicada, es decir, los subproyectos como JDBC (Java Database Connectivity) son específicos para esta función, también para web service o MVC. Es atómico en sus funciones, convirtiéndolo en reutilizable en el desarrollo en cualquier fase.

Spring basa toda su filosofía en estos tres principios que observamos en la siguiente figura 3.5

Inyección de dependencias (o Inversión de control)

El objetivo es conseguir un bajo acoplamiento entre los objetos de la aplicación. Con este patrón de diseño, los objetos no crean o buscan sus dependencias, estas son dadas al objeto. El contenedor es el encargado de realizar este trabajo en el momento de instanciar el objeto. Se invierte la responsabilidad en cuanto a la forma en que un objeto obtiene la referencia al otro objeto.

De esta forma, los objetos conocen sus dependencias para su interfaz. Así la dependencia puede ser intercambiada por distintas implementaciones a través del contenedor. En resumen, se programa orientado a interfaces y se inyectan las implementaciones a través del contenedor.

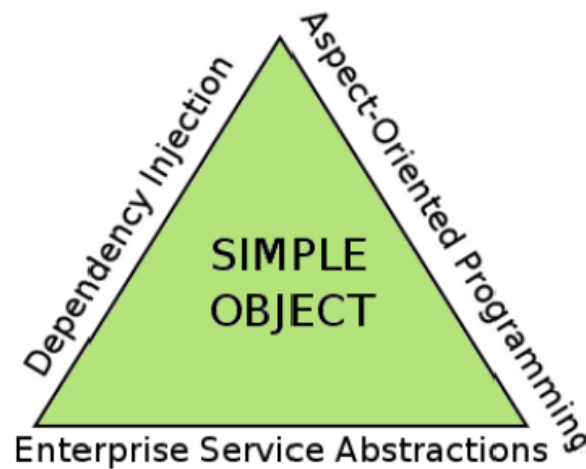


Figura 3.5: Principios Spring

Programación orientada a aspectos

Se trata de un paradigma de programación que intenta separar las funcionalidades secundarias de la lógica de negocio. Por ejemplo, la seguridad, la gestión de transacciones, etc. Estas son funcionalidades que atraviesan el programa en distintas abstracciones de este. Por tanto se corre el riesgo de caer en la repetición de código y el acoplamiento entre la lógica de negocio y la implementación de las funcionalidades transversales a las aplicaciones.

Al trabajar con AOP (Aspect-Oriented-Programming) se está centralizando el código que implementa estas funcionalidades y configurando Spring para que las ejecute donde se desee, en lugar de tener que hacerlo en cada momento que se haga uso de estas.

Abstracción de servicios de la empresa

Spring se encarga de hacer transparente muchas tareas que hay que realizar al trabajar con servicios utilizados habitualmente en aplicaciones empresariales y que suelen resultar muy repetitivas. Por ejemplo, acceder a la base de datos mediante JDBC, entonces Spring se encarga de gestionar las conexión, unificar excepciones, etc.

Los módulos que integren Spring pueden trabajar independientemente unos de los otros. Spring está formado más o menos por unos 20 módulos. Ahora se resumen los más importantes.

La parte fundamental de Spring es su módulo Core. Este se encarga de la inyección de dependencias. El concepto de este módulo es el BeanFactory, que implementa el patrón de diseño Factory. Encima del Core está el módulo Context que se encarga de ofrecer las herramientas para acceder a los beans.

- DAO: proporciona una capa de abstracción sobre JDBC, abstrae su código de acceso a datos de una forma simple. Cuenta con una capa de excepciones sobre los mensajes de errores proporcionados por cada servidor específico de base de datos.
- AOP: esta capa desacopla el código de una forma limpia, implementando funcionalidades que por lógica y claridad deberían estar separadas. Usando metadatos a nivel de código fuente se pueden añadir diversos tipos de información y comportamiento al código.
- ORM: ofrece capas de integración para APIs de mapeo de objetos, incluyendo JDO (Java Data Objects), Hibernate, y myBatis. Usando el paquete ORM es posible usar estos

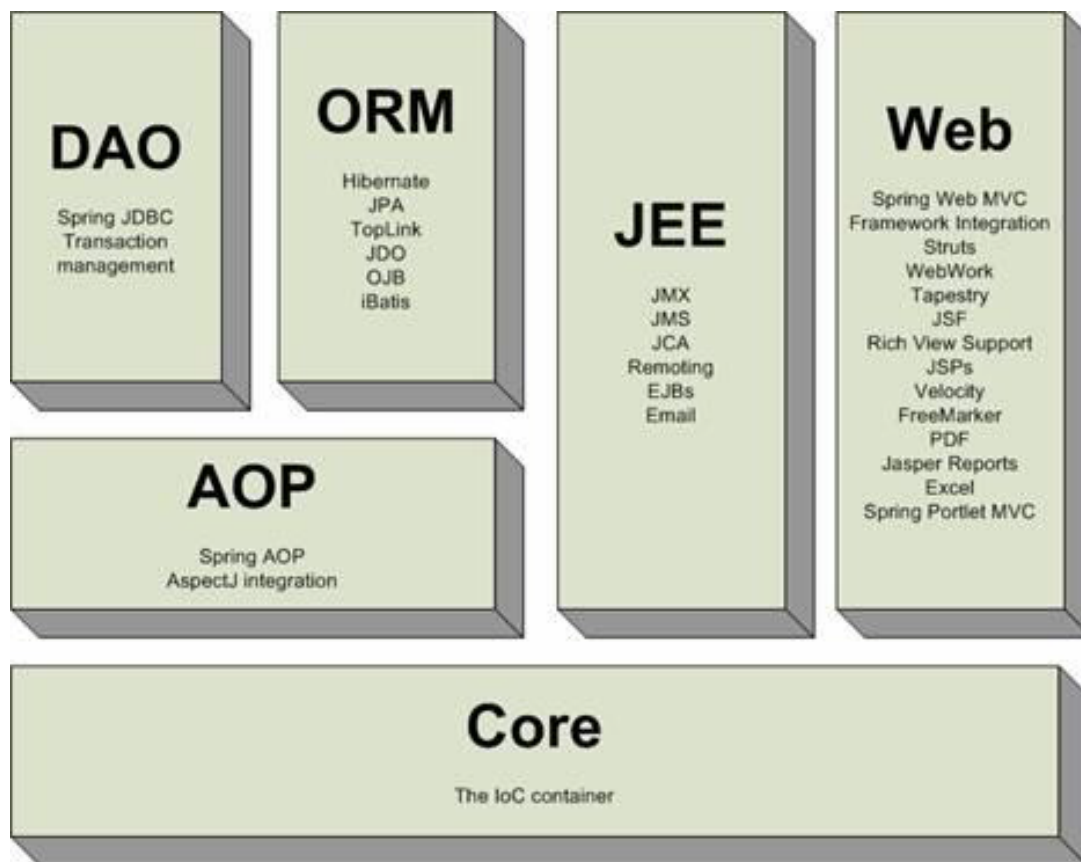


Figura 3.6: Módulos que integran el framework Spring

mapeadores conjuntamente con otras características que ofrece Spring.

- **Web**: proporciona características básicas de integración orientadas a la web, como la inicialización de contextos mediante servlet listener o un contexto de aplicaciones orientado a la web. Cuando se usa Spring conjuntamente con Struts, este es el paquete que permite una integración sencilla.
- **J2EE**: ofrece integración con aplicaciones Java Enterprise Edition, así como, servicios JMX (Java Management eXtensions), JMS (Java Message Service), EJB (Enterprise Java Bean), etc.

Spring MVC

Spring MVC es uno de los módulos del framework Spring, y este ofrece un exhaustivo soporte para el patrón MVC, así como, también ofrece soporte de otras características. Una de estas es facilitar la implementación de la capa de presentación. Spring proporciona un MVC para web bastante flexible y configurable, pero eso no le quita sencillez ya que puede realizar aplicaciones sin tener que configurar muchas opciones.

El MVC de Spring tiene algunas características principales:

- Clara división entre controladores, modelos web y vistas
- Basado en interfaces y flexible
- Ofrece interceptores al igual que controladores
- No obliga a usar JSP como única tecnología en la capa de vista
- Controladores son configurados como los otros objetos, mediante de IoC

En Spring MVC, nuestro DispatcherServlet funciona sobre el patrón Front Controller. El patrón Front Controller nos da un punto de entrada único a nuestras peticiones. De forma que todas ellas van a pasar por un mismo servlet, en el caso de Spring MVC, se trata de DispatcherServlet. Este servlet se encarga de gestionar toda la lógica de nuestra aplicación.

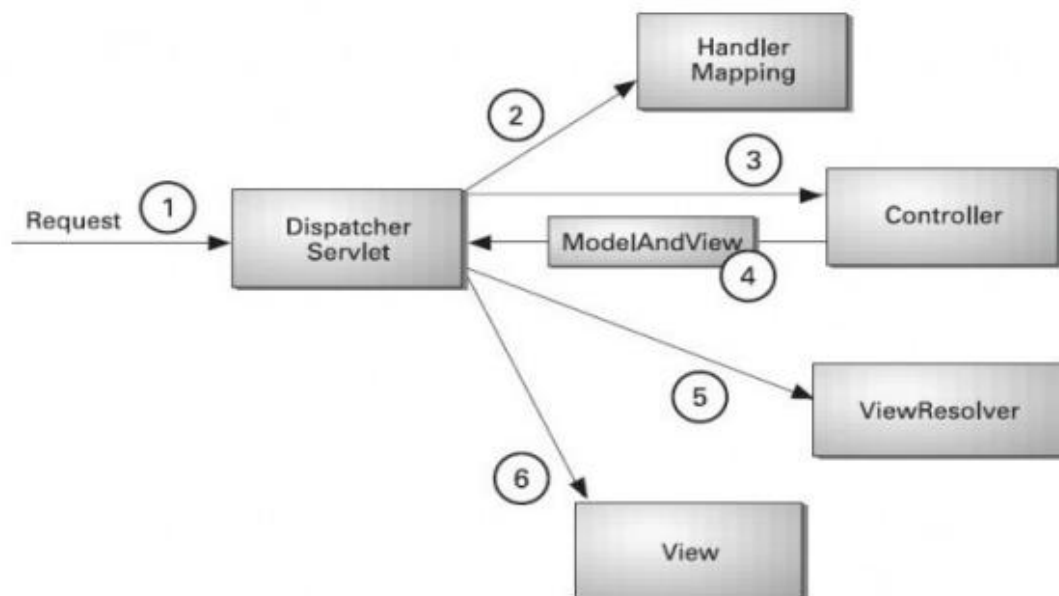


Figura 3.7: Funcionalidad básica en una aplicación Spring MVC

El flujo básico de una aplicación sobre Spring MVC según observamos en la figura 3.7 es el siguiente:

- La petición llega al DispatcherServlet (1)
- El DispatcherServlet tendrá que encontrar qué controlador va a gestionar la petición. Se realiza en la fase HandlerMapping (2)
- Localizado el Controller, el DispatcherServlet dejará la gestión al Controller de la petición (3). Al controlador se realiza toda la lógica de negocio de la aplicación, es decir, aquí se llama a la capa de servicios. El controlador retorna al Dispatcher un objeto de tipo ModelAndView (4). Donde el Modelo serán valores que se obtienen de la capa de servicio y View el nombre de la vista en la que se quiere mostrar la información que va dentro del Modelo. Una vez pasado el ModelAndView al DispatcherServlet, será este el que tendrá que asociar el nombre de la vista retornada por el controlador a una vista concreta (pagina JSP, JSF, etc). Este proceso viene indicado en la figura como ViewResolver (4).

- Finalmente y una vez resuelta la vista, el DispatcherServlet tendrá que pasar el Modelo a la vista concreta View (5).
- En los pasos de HandlerMapping, selección de Controller y ViewResolver se puede indicarle al DispatcherServlet qué estrategia seguir.

Resumen

- Ventajas
 - Ofrece división clara entre Controllers, Models, y Views
 - No obliga a usar JSP. Permite usar XSLT, Velocity, FreeMaker o implementar tu propio lenguaje
 - Muy flexible: implementación mediante interfaces
 - Los controladores se configuran mediante IoC como los otros objetos (fácilmente testeables y integrables)
 - Ofrece un framework para todas las capas de la aplicación
 - Se enlaza directamente con los beans de negocio, no existen ActionForms
 - Cantidad de código testable, las validaciones no dependen de la API de servlets
- Inconvenientes
 - Configuración compleja: muchos archivos XML
 - Algunos momentos demasiado flexible: no dispone de un Controlador común
 - Curva de aprendizaje larga

Informe de uso de frameworks J2EE

Se ha realizado un informe que ha elaborado la gente de ZeroTurnAround que puede ser muy útil a la hora de tomar decisiones sobre el framework a escoger. Este informe es muy completo y exhaustivo, analiza los frameworks Spring MVC, Grails, Vaadin, GWT, Wicket, Play, Struts y JSf. El informe ha estado realizado en 2012 y se basa en 1800 responsables de desarrollo.

Se comparan las 8 siguientes categorías vistas en la figura 3.8.

El gráfico 3.9 muestra las puntuaciones obtenidas por cada framework, basándose en las características anteriores vistas en la figura 3.8.

1. Rapid application prototyping
2. Framework Complexity
3. Ease of Use
4. Documentation & Community
5. Framework Ecosystem
6. Throughput/Scalability
7. Code Maintenance/Updates
8. UX, Look and feel

Figura 3.8: Categorías del análisis de frameworks

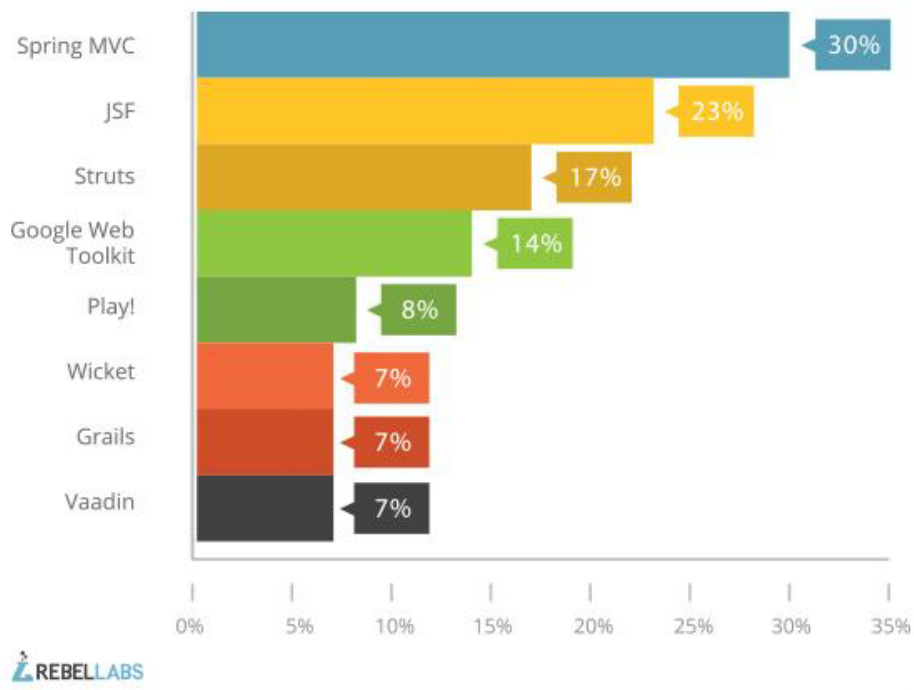


Figura 3.9: Porcentajes informe frameworks

3.4.2. Framework de persistencia

Hibernate

Hibernate ha sido la elección para ser el framework de persistencia. Su lugar está entre la capa de código de la aplicación Java y la de la API JDBC (Java Database Connectivity). Facilita la comunicación con la base de datos y el mantenimiento de la información relevante para la lógica de negocio.

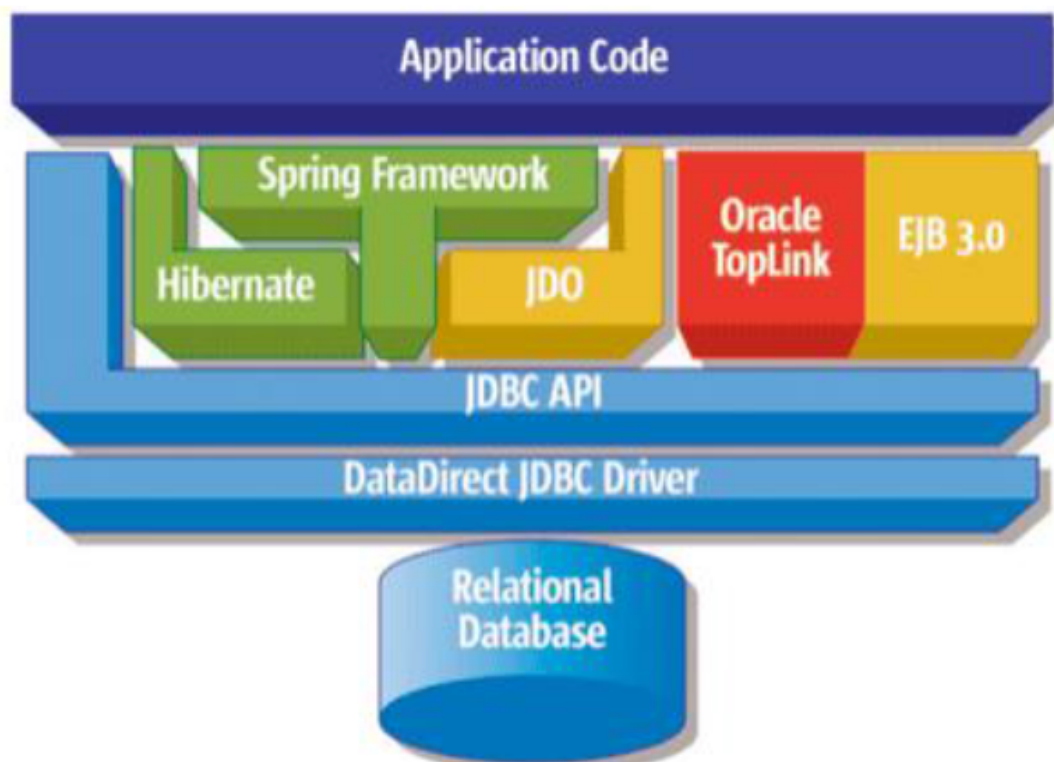


Figura 3.10: Arquitectura de la relación entre persistencia y base de datos

Hibernate es una herramienta ORM completa que ha conseguido en un tiempo récord una excelente reputación en la comunidad de desarrollo y se posiciona claramente como el producto Open Source líder de este campo gracias a sus prestaciones, buena documentación y escalabilidad.

Hibernate parte de una filosofía de mapeo de objetos Java, también conocido como POJOs (Plain Old Java Objects). No contempla la posibilidad de automatizar directamente la persistencia de Entity Bean tipos BMP (es decir, generar automáticamente este tipo de objetos), aunque sí que es posible combinar Hibernate con este tipo de beans utilizando los patrones conocidos para la delegación de persistencia en POJOs.

A continuación se mencionan algunas de sus características principales:

- **Simplicidad y flexibilidad:**
Necesita de un único archivo de configuración en tiempo de ejecución y un documento de mapeo para cada aplicación. Este archivo puede ser el estándar de Java o un fichero XML. El uso de frameworks de persistencia, como ahora EJBs hace que la aplicación dependa del framework. Hibernate no crea esta dependencia adicional. Los objetos persistentes en la aplicación no tienen que heredar de una clase de Hibernate o obedecer a una semántica específica. Tampoco necesita de un contenedor para funcionar.
- **Prestaciones:**
Una de las grande confusiones que aparecen al usar este tipo de frameworks es creer que las prestaciones se resentirán. Este no es el caso de Hibernate. Muchos frameworks de persistencia actualizan los datos de los objetos incluso cuando no han cambiado su estado. La caché de objetos juega un papel muy importante en la mejora de las prestaciones de la aplicación. Hibernate acepta diferentes productos en la caché, tanto de código abierto como comercial.
- **Completo:**
Ofrece todas las características de la orientación a objetos, incluyendo la herencia, tipos de usuario y las colecciones. También proporciona una capa de abstracción SQL nombrada HQL. Las sentencias HQL son compiladas por el framework de Hibernate y puestas en la caché para su posible reutilización.

La figura 3.11 muestra los roles de las interfaces más importantes en las capas de persistencia y de negocio de un aplicación J2EE. La capa de negocio está situada sobre la capa de persistencia, ya que la capa de negocio actúa como un cliente de la capa de persistencia. Además, Hibernate hace uso de Apis de Java, como ahora JDBC, JTA (Java Transaction Api) y JNDI (Java Naming Directory Interface).

Resumen

- **Ventajas**
 - Potente lenguaje de consulta (HLQ)
 - Transacciones, cache, asociaciones, polimorfismo, herencia, lazy loading, persistencia transitiva, etc.
 - Mucha documentación
 - No intrusivo
 - Comunidad activa con muchos usuarios
- **Inconvenientes**
 - Mayor complejidad de diseño
 - Curva de aprendizaje lenta
 - Más sobrecarga que las consultas directas SQL
 - No apta para aplicaciones de gran gestión de datos

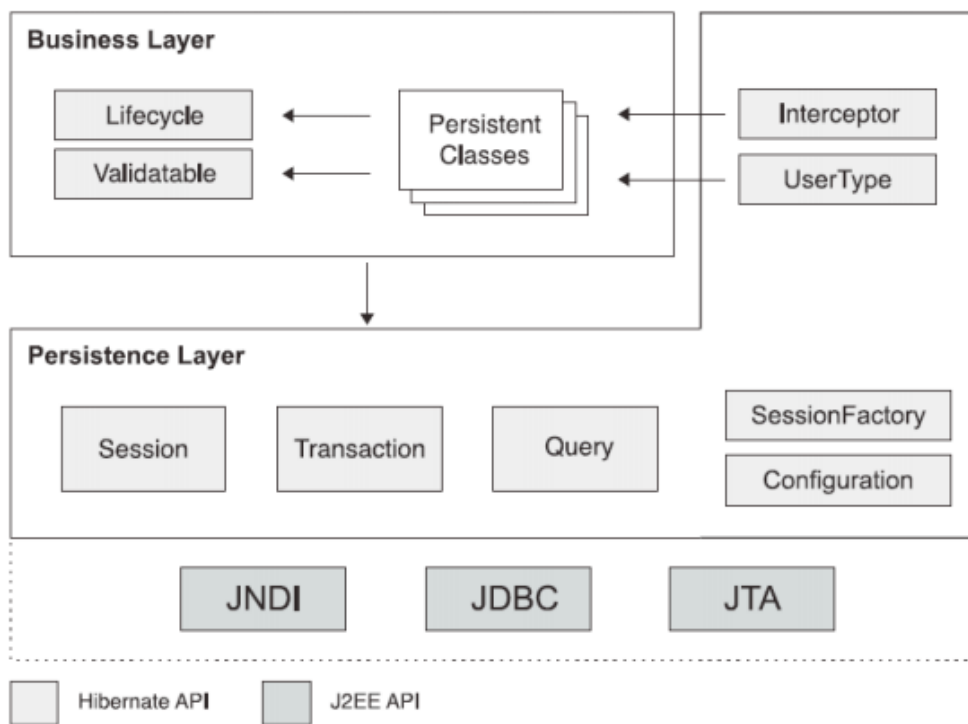


Figura 3.11: Roles de las interfaces Hibernate entre capas de persistencia y lógica de negocio

Capítulo 4

Arquitectura Liferay

En este apartado se va a definir el gestor de contenidos (CMS -Content Management System) Liferay a nivel global, donde se va a dar una visión general de cómo funciona el portal, de sus características principales y de sus componentes.

En segundo lugar, se detallan los diferentes componentes sobre los que se ha trabajado en el prototipo.

Finalmente se muestra el entorno de desarrollo para que se pueda ver a grandes rasgos cómo se enlazan todas las tecnología y componentes.

4.1. Introducción Portal Liferay

Se necesita paliar el problema de la descentralización de información buscando un portal web como un único punto de acceso a la información. Liferay Portal es una herramienta de gestión de contenidos, líder en la comunidad Java, con una orientación Web 2.0 y usa licencia Open Source. Es un gran gestor de contenidos ya que tiene una arquitectura flexible y modular, tiene multitud de herramientas para la gestión de contenidos y ahorra tiempo y recursos en crear páginas.

Algunas de las características principales:

- Reconocimiento internacional como la plataforma de portales más segura del mercado
- Funciona en todos los sistemas operativos, servidores de aplicaciones, base de datos
- Basado en estándares: JSR 127(JSF), JSR 168 (Portlet Specification), JSR 286 (Portlet 2.0 Specification), JSR 170 (Content Repository), JSR 208 (Java Business Integration), Ajax, Spring, Struts, Tiles, Velocity, WSRP
- Potente sistema de gestión de organizaciones, usuarios y roles
- Incorpora una suite de herramientas colaborativas (wiki, calendario, blogs, foros, etc)
- Autenticación y Single Sign-On (SSO): LDAP, Facebook, OpenID, etc.

- Integración con sistemas de terceros como el gestor documental Alfresco

En la figura 4.1 se puede ver todo lo que engloba la Arquitectura Lógica de Liferay:

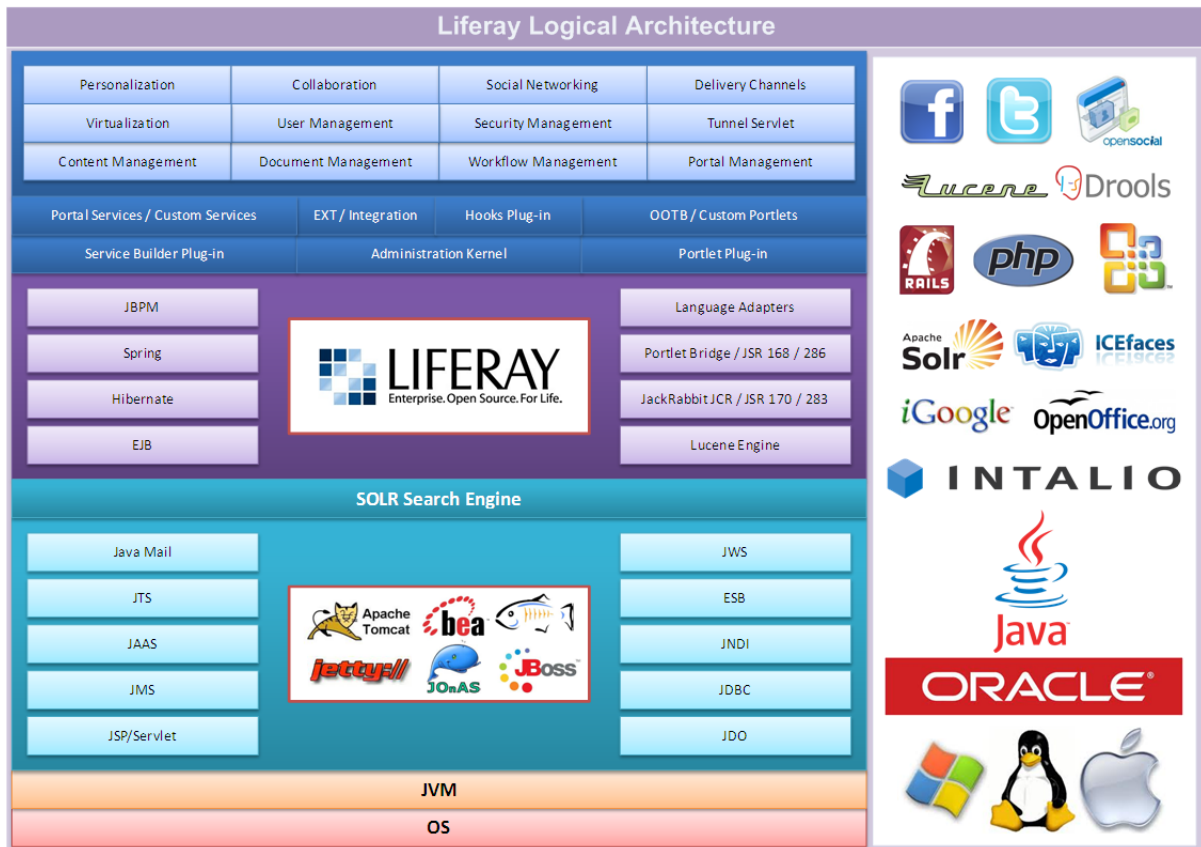


Figura 4.1: Arquitectura lógica de Liferay

4.2. Arquitectura de los componentes

En Liferay intervienen muchos tipos de componentes como vemos en la figura 4.2 donde se muestra la arquitectura de componentes que tiene:

Se hace hincapié sobretodo en los plugins, ya que es la parte que se usa para desarrollar una parte de la aplicación. Existen 4 tipos de plugins:

Theme

Un Theme o Tema de apariencia es un plugin que permite modificar el estilo o diseño de un portal web o de una página concreta. Cada Tema de apariencia puede disponer de una serie de esquemas de color que permiten hacer diversas variantes del mismo Tema de Apariencia y mantenerlo todo en un mismo plugin. Esto permite diferenciar las secciones usando distintos colores.

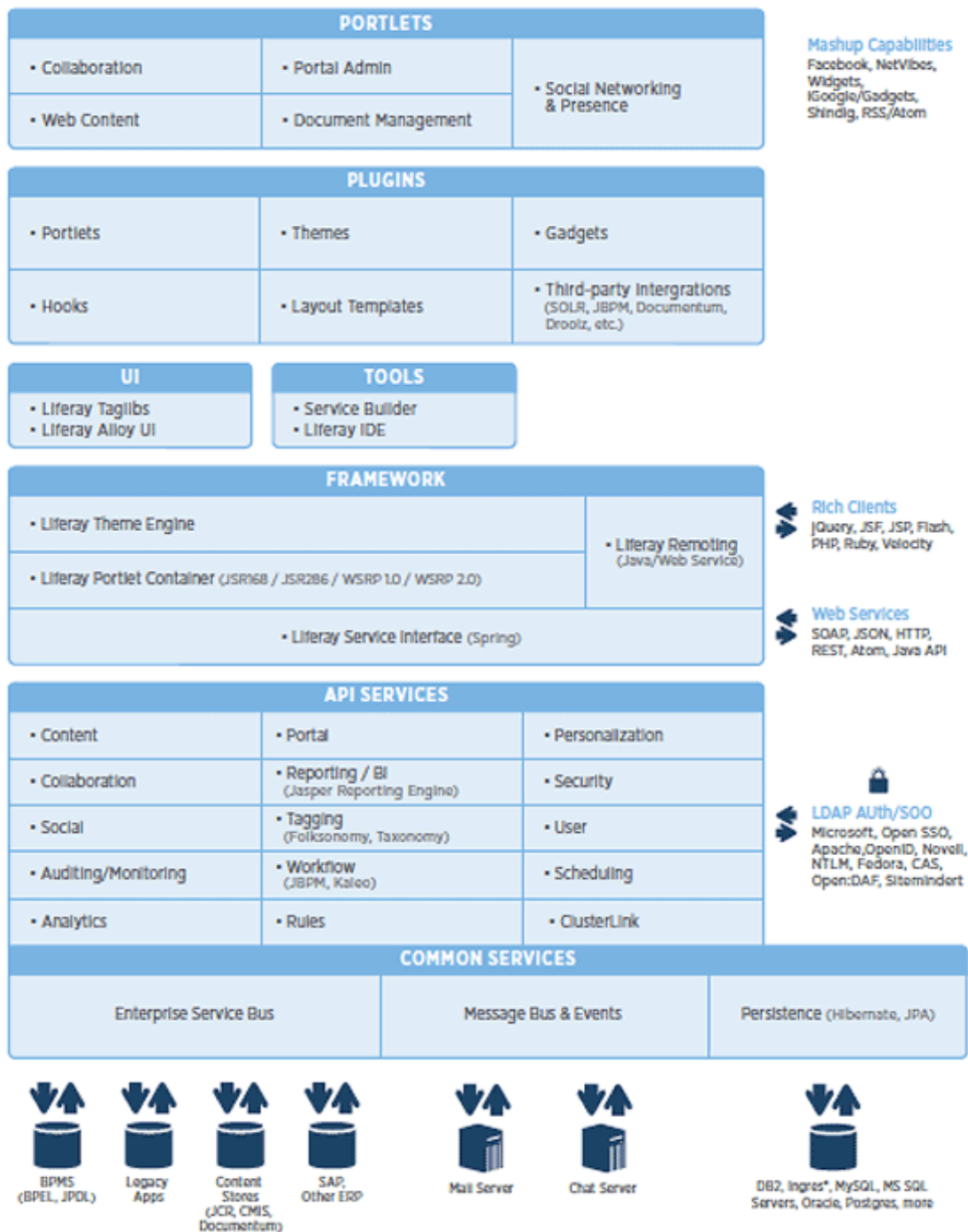


Figura 4.2: Arquitectura de los componentes que dispone Liferay

Layout

Layout o Plantilla es un plugin que permite definir una disposición de página diferente para cada una de las páginas. Son sobre estas las que se colocan los portlets y afecta única y exclusivamente al contenido central de la página. La cabecera, navegación principal y footer

se definirán directamente en el theme. Por ejemplo, se puede tener plantillas de 2 columnas al 40 %, 3 columnas al 33 %, 1 fila al 100 %,etc.

Hook

A diferencia de los portlets, este tipo de plugin permite modificar el código nativo del portal. Se utilizan para modificar el código de un portlet nativo del portal, definir variables de idioma o modificar alguna funcionalidad del portal.

Existen 4 tipos de hooks principales:

- Custom JSPs: Permiten modificar el código de un o diversos portlets mediante la modificación de sus JSPs
- Portal properties: Permiten modificar las propiedades del portal o definir nuevas
- Services: Permiten modificar los servicios del portal o directamente añadirse de nuevas
- Language properties: Permiten declarar nuevas variables de idiomas para facilitar la internalización del portal. Estas variables después serán usadas en distintos puntos del portal para traducciones.

Portlet

Los portlets son el componente principal de programación de Liferay. Son componentes modulares de una interfaz de usuario que proporciona contenido específico, el cual puede ser un servicio o datos provisionados por un sistema de información. Este se genera mediante el lenguaje de programación orientado a objetos Java y JSP, el cual presenta su contenido a los usuarios del portal, que es el sistema donde estos son ejecutados y se visualizan como una colección de ventanas de portlet que no se solapan, donde cada ventana muestra un portlet.

Estos portlets son componentes modulares, ya que cada uno de ellos se encarga de generar su propia interfaz de usuario del portal, ya que es independiente de esta, y a causa de esto es un componente portable. Por tanto, funcionan como módulos independientes y cada uno tiene una funcionalidad diferente. Cualquier código creado dentro de un portlet no afectará nunca al código nativo del portal.

Como ya se ha comentado anteriormente Liferay dispone de muchos portlets nativos para ser usados, pero también se puede crear un portlet nuevo mediante los estándares establecidos. Por ejemplo, el estándar JSF-2864, que será escogido para desarrollar el portlet de este proyecto. Podemos destacar dos fases principales que tiene y observamos en la figura 4.3:

- Render: se llama cuando el portlet necesita pintar la página, es decir, la acción se hace para cargar la página
- Action Phase: es cuando el JSP del portlet llama a una 'ActionURL', hace el proceso asignado a esta 'ActionURL' y después vuelve a hacer la fase render.

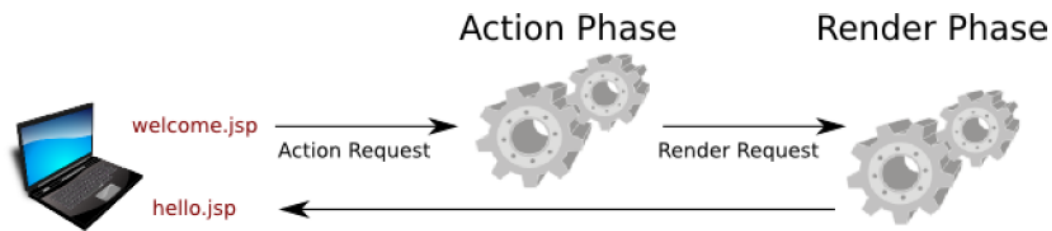


Figura 4.3: Flujo básico entre las páginas JSP y las fases principales

Una vez vistas las fases que puede tener un portlet, es importante conocer también su ciclo de vida que consta de cuatro fases y observamos en la figura 4.4:

- *init()*: método para inicializar el portlet. Se llama una sola vez en su creación
- *processAction()*: método llamado cada vez que se realiza una acción (o ActionURL) sobre el portlet
- *render()*: método para realizar el renderizado (o pintado) del portlet
- *destroy()*: método que se ejecuta solo una vez, al eliminar el portlet. Básicamente sirve para liberar posibles recursos que utilice el portlet

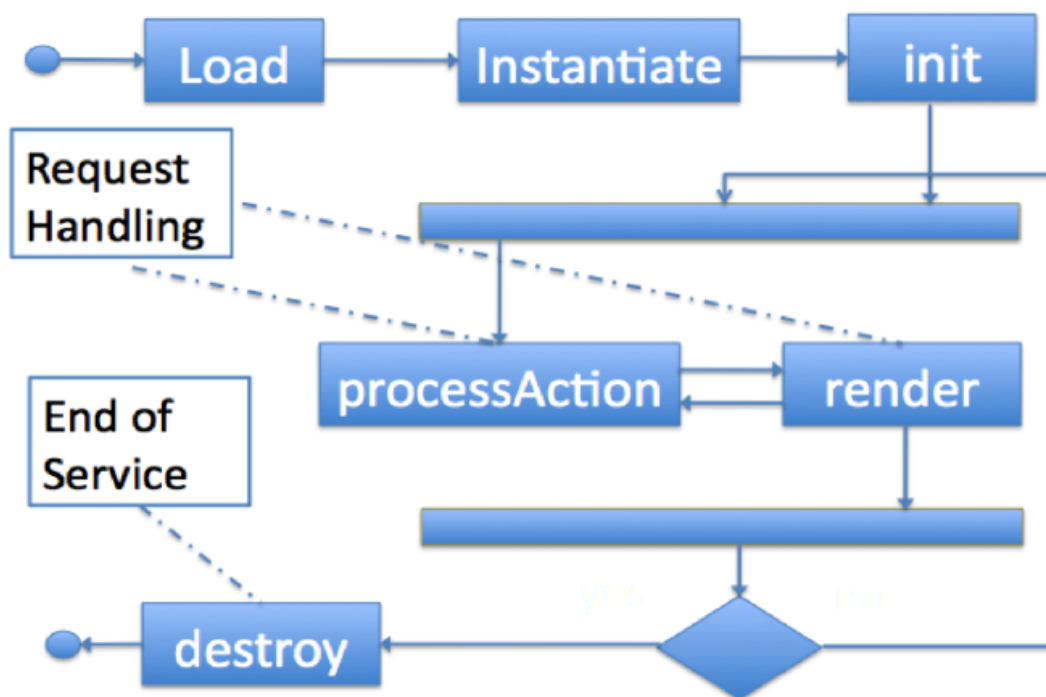


Figura 4.4: Ciclo de vida de un portlet.

4.3. Entorno

Vistos los principales componentes que intervienen en Liferay, se expone en el entorno que necesita Liferay para ser desplegado.

En la figura 4.5 se muestra el servidor Liferay en un entorno corporativo, situado a la LAN (Local Area Network) interna y intercomunicado con todos los otros sistemas para que sus componentes puedan interactuar con todos los elementos y servicios.

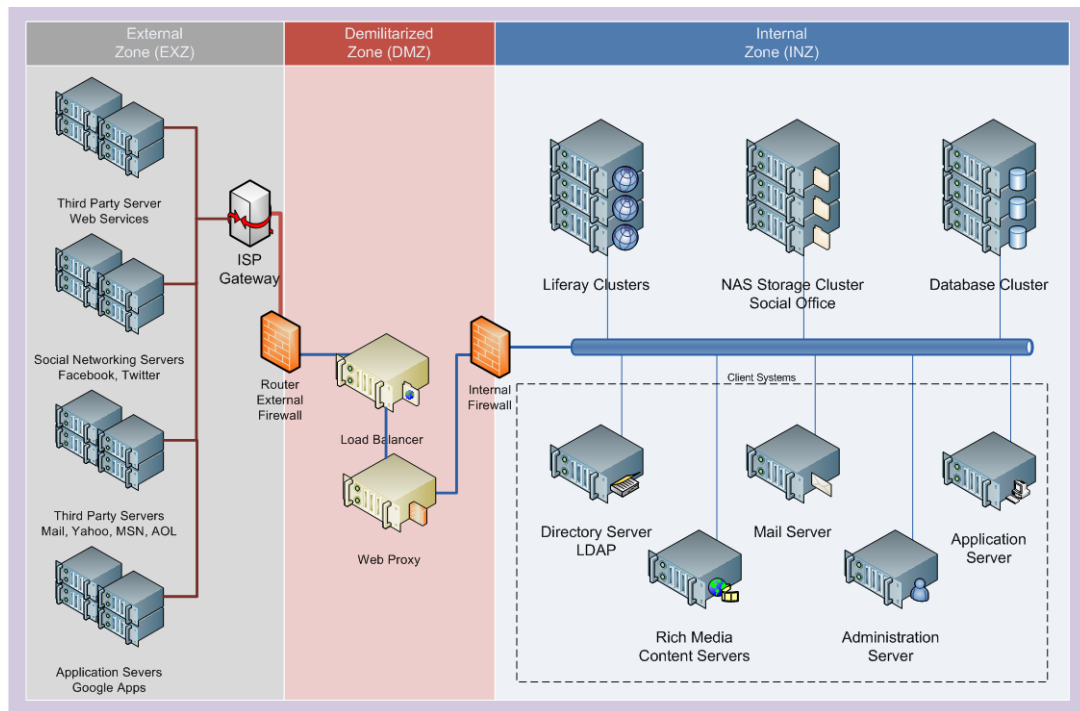


Figura 4.5: Entorno Liferay.

Liferay destaca por su compatibilidad y fácil integración con los sistemas, tecnologías y herramientas más populares del mercado de hoy en día.

Resumen de las características de compatibilidad:

- Sistemas operativos
 - Linux (CEntOS, RHES, SUSE, Ubuntu y otros)
 - Unix (AIX, HP-UX, Mac OS X, Solaris y otros)
 - Windows
- Contenedores de Servlets
 - Jetty
 - Resin
 - Tomcat
- Servidores de aplicaciones

- Geronimo
- GlassFish
- JBoss
- OracleAS
- WebLogic
- WebSphere
- Java Runtimes
 - Java Standard & Enterprise Edition (SE/EE) 5
 - Java Standard & Enterprise Edition (SE/EE) 6
 - Java Standard & Enterprise Edition (SE/EE) 7
- Base de datos
 - IBM DB2
 - MySQL
 - Oracle
 - PostgreSQL
 - SQL Server
- Entornos de Cloud Computing
 - Liferay está preparado para ser desplegado en la nube y entornos virtualizados, incluyendo EC2 y VMWare

Por último, hay que destacar que Liferay ha recogido en los últimos años premios y reconocimientos, no solo por ser el mejor entorno Open Source para portales, sino por su característica innovadora y visionaria, situándose como Leader en el denominado cuadrante mágico de Gartner como se ve en la figura 4.6.

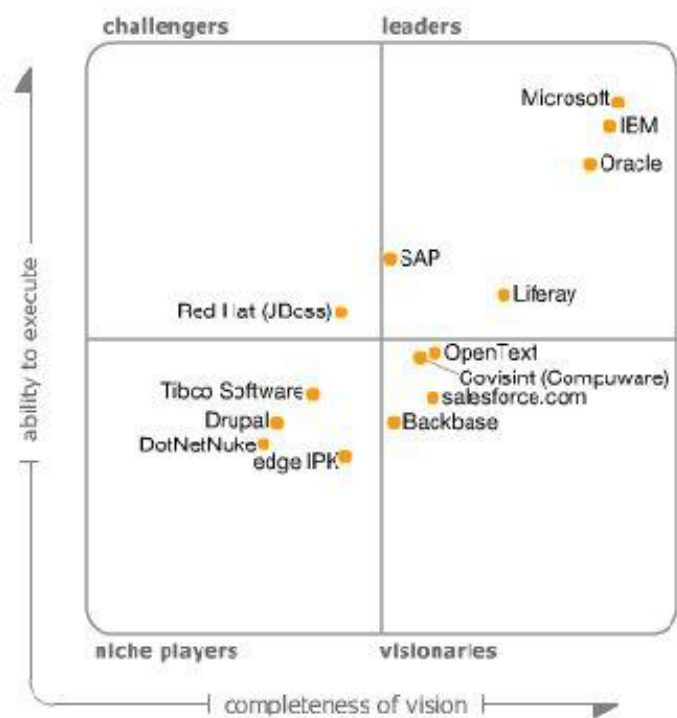


Figura 4.6: Cuadrante de Gartner

Capítulo 5

Análisis funcional

Durante el análisis, se recorren las distintas fases que se observan en la figura 5.1 hasta obtener las pruebas de aceptación necesarias para realizar la implementación mediante la metodología de desarrollo guiado por pruebas (ATDD).

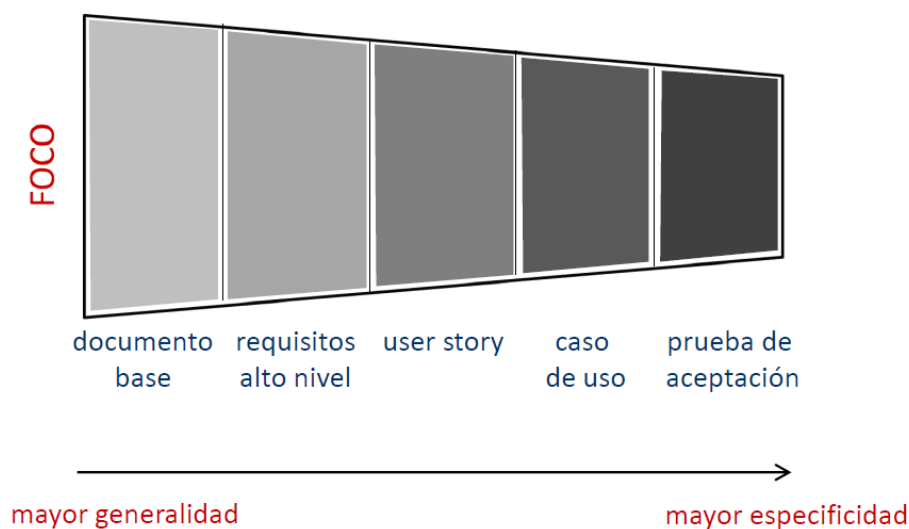


Figura 5.1: Pasos para obtener pruebas de aceptación

Primero se define el proyecto de forma general, mediante un documento base. Particularmente, se ha adaptado esta fase creando un 'mapa web' del futuro portal. Se ha decidido hacer de esta forma puesto que, para los implicados, era más fácil ver y entender la envergadura de la definición.

Con la estructura definida a grandes rasgos, se da lugar a definir los requisitos del proyecto. Los requisitos son las restricciones, objetivos o comportamientos que se espera del portal. Por ejemplo, un requisito sería que 'Los usuarios han de identificarse para poder comentar las noticias'.

Se observa en esta frase que las noticias se pueden comentar, aunque, con la condición que el usuario esté identificado.

De un requisito se pueden obtener distintas interpretaciones según roles y permisos del usuario que lo utiliza. Por ello, de cada requisito se obtienen una o más historias de usuario. Las historias de usuario son frases cortas que determinan que acción va a hacer un rol de usuario en concreto y que espera obtener con ello. Veamos un ejemplo:

Como: Administrador

Quiero: establecer el período de expiración de la contraseña de los usuarios

Para: que los usuarios se vean forzados a cambiar sus contraseñas periódicamente.

Se entiende que el rol es el de administrador, lo que quiere y para que lo quiere. Estas historias de usuario han de cumplir con el acrónimo INVEST acuñado por Bill Wake para referirse a ciertas propiedades que deberían tener una buena historia de usuario:

- Independiente
- Negociable
- Valiosa
- Estimable
- Sucinta

De las historias de usuario se obtienen los casos de uso. Estos describen los pasos que ha de seguir la acción que describe la historia de usuario para realizarse. Determina las condiciones que se han de dar antes y después de realizar la acción. Así como, los pasos que se dan en la interacción del usuario con el sistema. También define las excepciones y caminos alternativos que se pueden dar.

Finalmente, se definen las pruebas de aceptación. Conjuntamente con el cliente y el equipo de desarrollo. Se plantean de cada caso, las pruebas de aceptación necesarias. El cliente aportará casos básicos y el equipo de desarrollo aportarán casos extremos. La estructura básica sigue tres principios como observamos a continuación:

Dado un estado inicial del sistema

Cuando se realice una acción determinada

Entonces se da un estado final que se verifica

5.1. Documento base

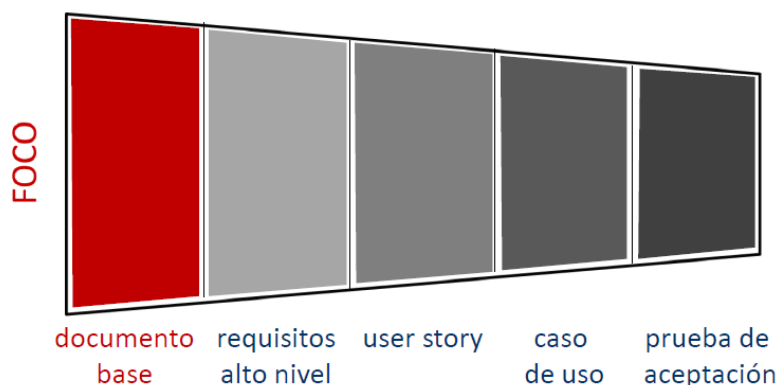


Figura 5.2: Fase 1. Documento base. Visión general del proyecto.

1. Portada

La portada es la primera página que se ve en el portal al acceder. Esta es neutra, muestra los contenidos destacados de todas las secciones del portal. Dichos contenidos pueden ser seleccionados por los responsables del portal, o programar de forma automática su aparición. En la parte superior está la cabecera con el logo, enlace a las tres grandes secciones del portal y herramientas comunes que faciliten la navegación como, un buscador, idiomas disponibles, el tiempo.

2. Gobierno Vinaros

Es la primera de las tres grandes secciones en que se divide el portal. En ella se muestra la parte más política de la corporación. Al entrar, se simula el comportamiento de la portada de la sección. Por lo tanto, se van a mostrar contenidos destacados de la sección en concreto.

2.1. Saludo del alcalde

Se observa la foto del actual alcalde. Junto a la foto reside un texto cordial de bienvenida y de introducción de las características de la ciudad.

2.2. Concejalías

Lista de las distintas concejalías y la foto del concejal actual. Al pulsar sobre un concejal, se abre una ventana donde se observa información del mismo.

2.3. Gobierno municipal

Esta subsección muestra información relacionada con los plenos municipales. Al entrar, se observa un calendario con las fechas de los plenos, pasados y futuros. También se puede ver una lista con los distintos videos de plenos pasado almacenados en Youtube.

2.3.1. Avisos, anuncios y convocatorias

Visible para todos los usuarios, en este punto se ofrece toda la información de los plenos. De forma organizada estará accesible la fecha de los mismos, junto a información relevante como los documentos oficiales en formato PDF.

2.3.2. Vídeos

Integración de vídeo en Streaming el día del pleno y una Videoteca con plenos pasados. Los plenos, tanto en directo, como los de la Videoteca gozan de total libertad. Son para todos los usuarios interesados en verlos.

2.3.3. Calendario de plenos y comisiones

Calendario donde se mostrarán las fechas de los plenos (pasados y futuros), así como, de las comisiones plenarias.

2.3.4. Centro de documentación

Aquí estará organizada toda la documentación relacionada con los plenos y las comisiones. La documentación se oferta en formato pdf exclusivamente y es accesible para todos los usuarios.

- Normativa de los plenos y comisiones -¿Listado con las normas de los plenos y las comisiones.
- Acuerdos adoptados en los plenos -¿Actas de los acuerdos adoptados en los plenos.
- Pregunta al pleno -¿Formulario para enviar pregunta que se realizará en el pleno si se considera oportuno por los responsables.
- Concejal 22 -¿Encuesta con diferentes preguntas para realizar una en el pleno siguiente.

2.3.5. Corporación actual (Todos los grupos municipales)

Gráfico circular con la distribución de los concejales según cada partido. Abajo se listan los miembros agrupados según el partido político al que pertenecen.

2.4. Gobierno abierto

Muestra el contenido por defecto del punto 2.4.1.

2.4.1. Que es? Que aporta?

Descripción y definición de gobierno abierto.

2.4.2. Participación ciudadana

Descripción y definición de la participación ciudadana.

2.4.2.1. Espacios de participación

- Consejos sectoriales: Documentación de los distintos consejos autorizados por el ayuntamiento ofertada en formato PDF.
 - Consejo de Cultura
 - Actas de reuniones
 - Acuerdos
 - Estatutos
 - Consejo de ...
 - Actas de reuniones

- Acuerdos
 - Estatutos
- Procesos participativos puntuales
 - Procesos participativos: Procesos propuestos por los responsables donde los ciudadanos pueden participar y ofrecer su opinión. Se detalla la fecha y lugar donde se realizará y ha de acudir el ciudadano que así lo crea conveniente.
 - Abiertos
 - Finalizados
 - Acuerdos y medidas
 - Audiencias públicas
 - Fechas
 - Quien la solicita?
 - Consultas ciudadanas
 - Calendario: con fechas para las consultas
 - Resultados: documentos pdf con los resultados obtenidos
 - Historial: historial de consultas donde se observan los documentos de los resultados
- Pleno popular
 - Convocatoria
 - Orden del día
 - Envía tu propuesta para el orden del día
- Otras acciones de participación
 - Banco del tiempo: Un banco de tiempo es un sistema de intercambio de servicios por tiempo. En él la unidad de intercambio no es el dinero habitual sino una medida de tiempo, por ejemplo el trabajo por hora. Es un sistema de intercambio de servicios por servicios o favores por favores.
 - Registro ciudadano: Registro externo al portal, para los ciudadanos que deseen participar en acciones de participación más restringidas. Formulario que se envía al responsable para verificar y enviar la autorización vía email.

2.4.2.2. Pregunta al concejal

Formulario para enviar directamente la pregunta al concejal.

2.4.2.3. Espacio de opinión y debate

- Espacio de opinión -¿noticias enviadas por los órganos acreditados
 - Partidos políticos
 - Asociaciones

- Foro: de carácter general donde los temas de debate son propuestos por el ayuntamiento. Sólo usuarios identificados pueden participar
- Encuestas: Sólo usuarios identificados pueden participar

2.4.2.4. Asociaciones de vecinos y entidades

- Agenda
- Contacto: información de contacto de las asociaciones
- Enlace a los espacio de opinión

2.4.2.5. Normativa y reglamentación Documentos PDF con las reglas que rigen los distintos apartados de participación ciudadana.

2.5. Servicio de prensa

Notas de prensa y noticias de actuaciones del ayuntamiento y el equipo de gobierno.

2.6. Boletines oficiales

Enlaces a los distintos boletines.

3. Gestiona Vinaròs

Segunda gran sección con información de la gestión y documentación del ayuntamiento. Tanto interna, como para realizar trámites.

3.1. Organización municipal (Departamentos)

Se listan los distintos departamento en los cuales se descompone el ayuntamiento. Cada departamento dispone de su espacio privado que simula el comportamiento de un portal web independiente. Solo tiene la limitación de respetar la imagen corporativa teniendo la cabecera común que se detalla más adelante en este documento. (Añadir referencia). En cuanto a contenido, organización y número de páginas, el departamento es libre por medio de su responsable de gestionar su espacio. Liferay, candidato firme a ser la herramienta usada ofrece la posibilidad deseada sin mayor coste.

- Secretaría
- Urbanismo
- Actividades
- Informática
- Policía
- Recursos Humanos
- Contratación
- Comercio

- Obras y servicios
- Medio ambiente
- Servicios sociales
- Cultura
- Biblioteca
- Escuela para adultos (EPA)
- Educación
- Escuela de arte
- Deportes
- Juventud
- Turismo
- Intervención
- Tesorería
- Mercado
- Archivo
- Oficina de Información y Atención al Ciudadano (OIAC)
- Normalización lingüística

3.2. Sede electrónica (Realización de trámites)

Enlace a la sede electrónica del ayuntamiento.

3.3. Documentación

Portada con el listado de las distintas tipologías de documentación.

3.3.1. Normativa (Ordenanza municipal)

Documentos en formato PDF.

3.3.2. Resoluciones judiciales

Documentos en formato PDF.

3.3.3. Acuerdos

Documentos en formato PDF.

3.3.4. Convenios

Documentos en formato PDF.

3.3.5. Trámites

Enlace a la sede electrónica del ayuntamiento.

3.3.6. Impresos

Documentos en formato PDF.

3.4. Perfil de contratante (Contratos administrativos)

Enlace a la página oficial de la Generalitat Valenciana donde se muestra las características de las empresas para trabajar con el ayuntamiento.

3.5. Becas y subvenciones

Lista de las distintas becas y subvenciones que ofrece el ayuntamiento.

3.6. Trabajo público

3.6.1. Ofertas abiertas

Listado de ofertas de trabajo.

3.6.2. Promoción interna

Ofertas de carácter interno.

3.6.3. Histórico de ofertas

Listado de todas las ofertas pasadas.

3.7. Cartografía (Calles)

Mapa de las calles con información correspondientes a los distintos edificios públicos.

3.7.1 Nomenclátor

Enlace a la web propia Nomenclátor.

3.8. Vía pública

Enlace a la aplicación propia destinada a gestionar incidencias en la vía pública como obras, accidentes, cortes de calles, eventos, etc.

3.9. Trabajo y empresa

Sección gestionada por el responsable del departamento de Agencia de Empleo y Desarrollo Local (AEDL)

4. Vive Vinaròs

Tercera gran sección del portal. Esta es la sección más lúdica. En ella se va a informar de eventos relacionados con fiestas, cultura, deportes, ocio, etc.

4.1. Agenda

Agenda con información de los actos y un calendario.

4.2. Cultura, Ocio, Deportes, Juventud, Normalización lingüística

Secciones de los departamentos citados en el título del apartado.

4.3. Turismo

Enlace a la web de turismo.

4.4. Calles (Lugares de interés)

Mapa de la ciudad con lugares de interés etiquetados.

4.5. Cómo llegar a Vinaros

Opciones para planificar rutas de llegada.

4.6. Fiestas

Listado de las fiestas del pueblo con descripción. Videos de Youtube, noticias con fotos y más detalles.

4.7. Carnaval

Página dedicada exclusivamente a la fiesta del carnaval con un contador regresivo. Calendario, actos, noticias...

4.8. Directorio telefónico

Teléfonos de interés para el ciudadano.

5. Sección general del web

Funcionalidades generales que se podrán ver en todas las secciones.

5.1. Buscador

Búsqueda interna de contenidos.

5.2. Noticias

Noticias escritas por responsables del ayuntamiento para informar a los ciudadanos.

5.3. El tiempo y calidad del aire

Información del tiempo y calidad del aire.

5.4. Contacto/Sugerencias

Formulario para contactar anónimamente con el ayuntamiento.

5.5. Suscripción al boletín oficial

Formulario para suscribirse y recibir, vía email, contenidos que el usuario selecciona de una lista de categorías ofrecida por el responsable del portal.

5.6. Perfil del usuario

Lugar privado del usuario para gestionar su información.

5.7. RSS

RSS son las siglas de Really Simple Syndication, un formato XML para syndicar o compartir contenido en la web. De esta forma se puede acceder a la información de formas alternativas.

5.8. Aviso legal

Aviso de las responsabilidades legales por parte de los responsables del portal.

5.9 Mapa web

Guía y estructura del portal web mostrado de forma jerárquica.

5.2. Requisitos funcionales

Funciones de alto nivel del software que contribuyen a lograr objetivos

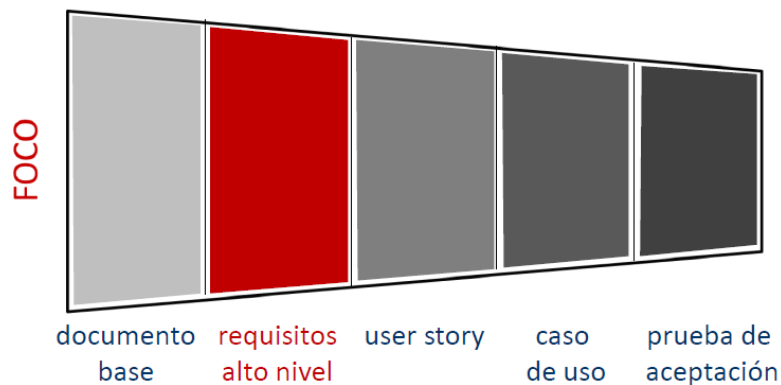


Figura 5.3: Fase 2. Requisitos de alto nivel.

Tipos de usuario: Se diferencian 4 grupos genéricos de usuarios.

- Administrador - Control total de la aplicación.
- Usuario del portal - Usuario sin ningún tipo de autenticación
- Usuario registrado - Usuario previamente registrado y que se ha autenticado.
- Usuario interno de la corporación - Usuarios que trabajan en la corporación y tienen funciones específicas de mantenimiento.

(Gestionar - Añadir, modificar y eliminar)

El sistema permite navegar por los contenidos vía web.

El sistema permite gestionar usuarios.

El sistema permite acceder a usuarios utilizando las credenciales del directorio activo de la corporación (Active Directory) mediante LDAP.

El sistema permite gestionar todas las entidades/información.

El sistema permite comentar y puntuar las noticias.

El sistema permite generar encuestas.

El sistema permite gestionar un foro.

El sistema permite compartir noticias y eventos en las redes sociales como, Facebook, Twitter, Google...

El sistema permite valorar cada página mediante una puntuación.

El sistema permite enviar un mensaje a los responsables del portal web.

El sistema permite modificar la estructura y organización del contenido en cada sección.

El sistema permite emitir flujos de vídeo en directo.

El sistema permite gestionar permisos sobre contenido del portal para limitar el acceso.

El sistema permite gestionar grupos de usuarios y roles.

El sistema permite acceder a la información mediante RSS (Really Simple Syndication).

El sistema permite subscribirse al boletín informativo para recibirlo vía email.
El sistema permite cambiar de idioma.

5.3. Historias de usuario

Cada requisito puede descomponerse en una o varias historias de usuario.

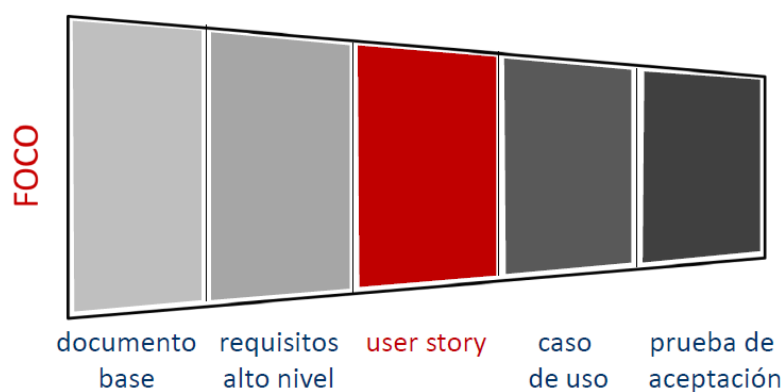


Figura 5.4: Fase 3. Historias de usuario.

El sistema permite navegar por los contenidos vía web.

Como usuario del portal, deseo acceder a la información sobre la que tengo permisos vía web para interactuar con ella y estar informado.

El sistema permite gestionar usuarios.

Como usuario del portal, quiero tener la posibilidad de registrarse en el sistema para tener almacenadas mis preferencias de navegación y participar en las acciones restringidas.

Como usuario registrado, quiero hacer cambios sobre mi información para mantener esta actualizada.

Como usuario registrado en el sistema, quiero poder dar de baja mis credenciales en caso de querer abandonar el uso del portal.

Como administrador, quiero poder gestionar la información de los usuarios para mantener orden y control sobre ellos por seguridad.

Como administrador, quiero poder bloquear usuarios que no respeten el buen uso del portal.

El sistema permite gestionar todas las entidades/información.

Como administrador, quiero poder añadir, modificar y eliminar la información del sistema según crea conveniente para poder mantener el portal actualizado y coherente.

Como administrador, quiero tener la posibilidad de validar la información antes de que esta sea publicada para verificar que cumple con las reglas de la corporación.

Como usuario interno de la corporación, quiero gestionar la información sobre la que tengo permisos para mantener mi sección actualizada.

El sistema permite comentar y puntuar las noticias.

Como usuario registrado, quiero comentar y puntuar la noticias para expresar mi opinión.

Como administrador, usuario interno de la corporación, quiero tener un control (gestionar) sobre los comentario de las noticias propias para verificar el buen uso del lenguaje.

El sistema permite generar encuestas.

Como administrador, usuario interno de la corporación, quiero poder generar encuestas para mostrarlas en mi sección y saber que opinan los usuarios.

Como administrador, usuario interno de la corporación, quiero poder generar informes de las encuestas propias para ver el nivel de aceptación.

(Solo usuarios identificados pueden responder a encuestas) Como usuario registrado, quiero responder a las encuestas para ofrecer mi opinión.

Como usuario registrado, quiero ver los porcentajes de respuestas en las encuestas para ver la opinión de la mayoría.

El sistema permite gestionar un foro.

Como administrador, quiero control total sobre el foro para mantener una estructura y orden coherentes.

Como usuario interno de la corporación, quiero añadir temas de discusión al foro para generar debates en mi sección.

Como usuario registrado, quiero comentar y puntuar discusiones del foro para ser partícipe mostrando mi opinión.

Como administrador, usuario interno de la corporación, quiero tener control sobre los comentarios para poder verificar el buen uso del lenguaje.

El sistema permite compartir activos del sistema en las redes sociales.

Como usuario del portal, quiero compartir el máximo contenido (noticias, eventos, pleno, vídeos) en mis redes sociales para mostrar a mis contactos lo que me parece relevante.

El sistema permite valorar cada página mediante una puntuación.

Como usuario registrado, quiero valorar las página mediante un sistema de puntuación para mostrar el grado de satisfacción.

El sistema permite enviar un mensaje a los responsables del portal web.

Como usuario del portal, quiero un formulario que me permita enviar un mensaje a los responsables del portal para transmitir mi opinión a quien pertoque.

Como administrador, quiero ver un listado de todas las opiniones para poder gestionarlas.

El sistema permite modificar la estructura del contenido en cada sección.

Como usuario interno de la corporación, quiero gestionar mi sección con total libertad para simular el comportamiento de un portal web independiente.

El sistema permite emitir flujos de vídeo en directo.

Como usuario del portal, quiero ver eventos del ayuntamiento en directo para estar al corriente de la actualidad.

Como administrador, quiero emitir eventos en directo mediante el portal web.

El sistema permite gestionar permisos sobre activos para limitar el acceso.

Como administrador, quiero otorgar permisos a los grupos de activos del portal para limitar el acceso y control sobre ellos.

El sistema permite gestionar grupos de usuarios y roles.

Como administrador, quiero generar una estructura de grupos de usuario y roles para asignar permisos dentro del portal y mantener su la seguridad.

El sistema permite acceder a la información mediante RSS.

Como usuario del portal, quiero recibir la información mediante RSS para estar actualizado al instante. El sistema permite suscribirse al boletín para recibir información vía email.

Como usuario del portal, quiero recibir el boletín informativo de las categorías que previamente he seleccionado como, noticias, eventos, plenos, fiestas para estar informado de todo lo que rodea el ayuntamiento.

El sistema permite cambiar de idioma.

Como usuario del portal, quiero cambiar de idioma para leer el contenido con el lenguaje que más se adapte a mis características.

5.4. Casos de uso

Una historia de usuario puede refinarse a través de use cases (casos de uso)
Un caso de uso describe una secuencia de interacción usuario - software
Se recomienda crear casos de uso sólo para la user story en desarrollo.

Se han definido uno pocos casos de uso que abarcan las funcionalidades más generales y que se utilizarán en la implementación.

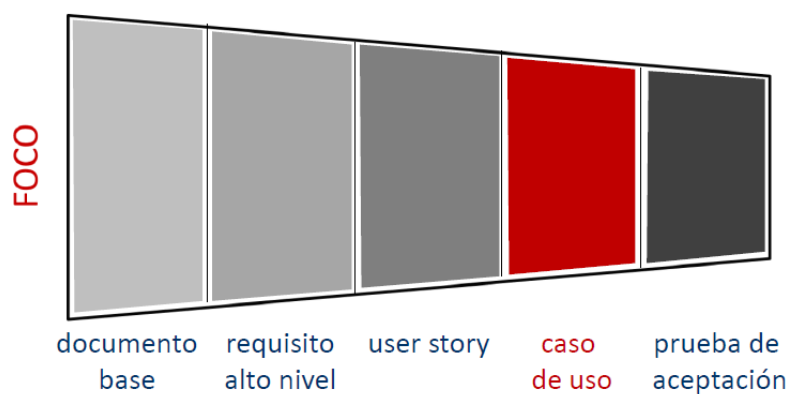


Figura 5.5: Fase 4. Casos de uso.

Roles:

- Usuario anónimo
- Usuario registrado
- Usuario interno de la corporación
- Administrador

Identificador: CU01

Nombre: Navegar vía web

Descripción: El usuario vía navegador web accede al contenido

Roles: Todos

Condiciones previas: El servidor está a la espera de peticiones

Post-condiciones: El usuario recibe la respuesta.

Flujo principal: (escenario principal)

1. El usuario solicita el contenido mediante una URL
 2. El sistema procesa la petición y genera una respuesta que envía al usuario
 3. El usuario recibe la respuesta
-

Excepciones: (escenario alternativo)

4. El usuario no tiene permisos para acceder al contenido
 - El sistema envía el mensaje de ?Acceso restringido?
 - El sistema redirige a la página de autenticación.
 5. El sistema no encuentra el recurso
 - El sistema envía como respuesta ?Contenido no encontrado.?
-

Identificador: CU02

Nombre: Registrar usuario

Descripción: Un usuario anónimo guarda sus credenciales en el sistema

Roles: Usuario anónimo

Condiciones previas: El usuario no está registrado en el sistema

Post-condiciones: El usuario queda almacenado en el sistema

Flujo principal: (escenario principal)

1. El usuario solicita registrarse
 2. El sistema responde con un formulario de registro
 3. El usuario rellena la información solicitada
 4. El sistema verifica que la información cumple los requisitos
 5. El sistema almacena las credenciales
 6. El sistema informa al usuario que se ha registrado correctamente
-

Excepciones: (escenario alternativo)

7. La información introducida no supera la verificación
 - El sistema retorna al punto 2 con la información introducida
 8. Las credenciales no se han podido almacenar
 - El sistema avisa al usuario del error
 - El sistema retorna al inicio del proceso
 9. Las credenciales ya existen en el sistema
 - El usuario es avisado y se le ofrece la posibilidad de recuperar la contraseña mediante un formulario
 - El usuario rellena el formulario y la contraseña es enviada vía email
-

Identificador: CU03

Nombre: Eliminar usuario

Descripción: Un usuario registrado es dado de baja

Roles: Usuario registrado

Condiciones previas: El usuario está logueado en el sistema

Post-condiciones: La cuenta queda suspendida

Flujo principal: (escenario principal)

1. El usuario se loguea en el sistema
2. El usuario solicita dar de baja su cuenta
3. El sistema solicita confirmación
4. El usuario responde a la confirmación
5. El sistema deja la cuenta en suspensión
6. El sistema envía al responsable un aviso para que confirme posteriormente la operación.
7. El sistema cierra la sesión del usuario

Excepciones: (escenario alternativo)

8. La sesión del usuario ha caducado
 - El sistema redirige a la página de login
 9. El usuario responde negativamente a la confirmación
 - El sistema redirige al usuario a su página de perfil
-

Identificador: CU04

Nombre: Gestionar información

Descripción: Intranet

Role: Administrador

Condiciones previas: Usuario logueado en el sistema

Post-condiciones: La información es actualizada

Flujo principal: (escenario principal)

1. El administrador solicita gestionar un tipo de información
 2. El sistema devuelve un listado con las últimas entradas y las opciones
 3. El administrador selecciona el registro y la opción
 4. El sistema genera el formulario
 5. El administrador rellena el formulario
 6. El sistema verifica la información
 7. El sistema actualiza la información
 8. El sistema responde con el nuevo registro y el mensaje correcto
-

Excepciones: (escenario alternativo)

9. La sesión ha caducado
 - El sistema redirige a la página de login
10. El usuario no tiene permisos para gestionar el tipo de contenido
 - El sistema informa al usuario
 - El sistema redirige a la página principal del panel
11. La información introducida no supera la validación
 - El sistema redirige al punto número 4 con el mensaje oportuno
12. La información no puede actualizarse en el sistema

- El sistema informa del error
 - El sistema redirige al punto 2
-

Identificador: CU05

Nombre: Estructura y organización de sección

Descripción: El responsable de sección puede manejar la estructura

Role: Usuario interno de la corporación

Condiciones previas: El usuario está logueado en el sistema

Post-condiciones: La estructura se almacena en el sistema

Flujo principal: (escenario principal)

1. El usuario solicita activar las herramientas de edición de estructura
 2. El sistema retorna al usuario la página con las herramientas activadas
 3. El usuario organiza el contenido y la estructura a su antojo
 4. El usuario pulsa el botón de almacenar y enviar la estructura al sistema
 5. El sistema almacena la estructura
-

Excepciones: (escenario alternativo)

6. La sesión ha caducado
 - El sistema redirige a la página de login
7. El usuario no tiene permisos para gestionar el tipo de contenido

- El sistema informa al usuario
- El sistema redirige a la página principal del panel

8. La información no puede actualizarse en el sistema

- El sistema informa del error
- El sistema redirige al punto 2

5.5. Pruebas de aceptación

Crear pruebas de aceptación a partir de cada escenario

Crear pruebas de forma conjunta: cliente + desarrollador/a + tester

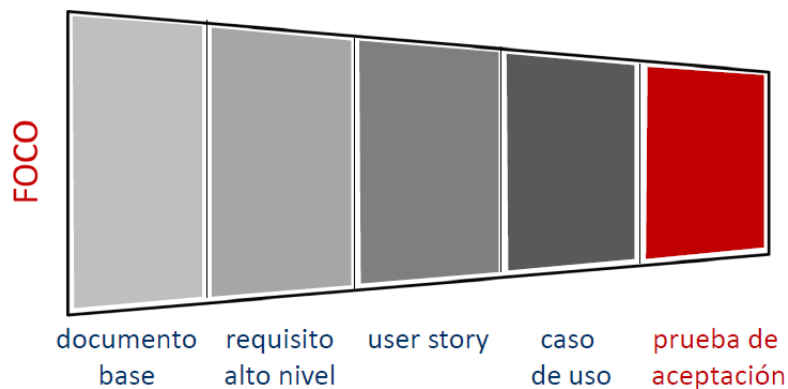


Figura 5.6: Fase 5. Pruebas de aceptación.

Se han definido pruebas de aceptación para la funcionalidad desarrolla. Aunque, en la práctica no se han implementado debido a malas prácticas propiciadas por la falta de tiempo y experiencia.

Dado / Given

Una opinión está almacenada en el sistema

Cuando /When

El administrador elimina la opinión

Entonces / Then

La opinión es borrada del sistema

Dado / Given

Una opinión está almacenada en el sistema

Cuando /When

El administrador modifica la opinión

Entonces / Then

La opinión es modificada

Dado / Given

Una opinión no está en el sistema

Cuando /When

El administrador añade la opinión

5.6. Requisitos de datos

Liferay ofrece gestión para la mayoría de los datos a almacenar, así que aprovechando su capacidad, sólo se documenta la información que no abarca Liferay.

Datos gestionados por Liferay:

- Usuarios
- Evento
- Video
- Secciones
- Categorías
- Documento
- Noticia
- Comentario
- Foro
- Encuestas
- Enlaces

Datos propios:

- Partido politico
- Concejalía
- Persona
- Departamento
- Consejo sectorial
- Entidad
- Proceso participativo
- Beca

Requisitos de datos

Identificador: RD01

Nombre: partido_politico

Autor: Dept. Informática

Fecha de creación: 03/09/2015

Fecha revisión: 10/09/2015

Versión: 1.0

Tipo: Datos a mantener

Información específica a salvar: nombre, logo, ano_fundacion, afiliados, escanos, web

Identificador: RD02

Nombre: concejalia

Autor: Dept. Informática

Fecha creación: 03/09/2015

Fecha revisión: 10/09/2015

Versión: 1.0

Tipo: Datos a mantener

Información específica a salvar: nombre, presidente, vocales, horario, telefono, presidente, fax

Identificador: RD03

Nombre: Persona

Autor: Dept. Informática

Fecha creación: 03/09/2015

Fecha revisión: 11/09/2015

Versión: 1.0

Tipo: Datos a mantener

Información específica a salvar: nombre, apellidos, DNI, partido_politico, comisiones, email

Identificador: RD04

Nombre: Departamento

Autor: Dept. Informática

Fecha creación: 03/09/2015

Fecha revisión: 11/09/2015

Versión: 1.0

Tipo: Datos a mantener

Información específica a salvar: nombre, telefono, email, nombre_seccion

Identificador: RD05

Nombre: consejo_sectorial

Autor: Dept. Informática

Fecha creación: 03/09/2015

Fecha revisión: 10/09/2015

Versión: 1.0

Tipo: Datos a mantener

Información específica a salvar: nombre, telefono, email, horario, ambito, participantes, fecha_creacion, logo, web

Identificador: RD06

Nombre: Entidad

Autor: Dept. Informática

Fecha creación: 03/09/2015

Fecha revisión: 10/09/2015

Versión: 1.0

Tipo: Datos a mantener

Información específica a salvar: nombre, logo, participaciones

Identificador: RD07

Nombre: Beca

Autor: Dept. Informática

Fecha creación: 03/09/2015

Fecha revisión:10/09/2015

Versión: 1.0

Tipo: Datos a mantener

Información específica a salvar: titulo, fecha_convocatoria, fecha_boe, fecha_semanario, fecha_dogv, fecha_instancia, documentos

Identificador: RD08

Nombre: Proceso participativo

Autor: Dept. Informática

Data creación: 03/09/2015

Data revisión: 11/09/2015

Versión: 1.0

Tipo: Datos a mantener

Información específica a salvar: titulo, resumen, explicacion, ubicacion, fecha_participacion, destinatarios, fecha_aplicacion

Capítulo 6

Diseño

6.1. Diseño arquitectónico

6.1.1. Entorno de producción

Una vez vistas las tecnologías a utilizar, se pasa a ver cómo será el entorno de producción de la aplicación.

El servidor Liferay va a estar situado en la LAN (Local Area Network) interna y intercomunicado con todos los otros sistemas, para que así, puedan interactuar sus componentes con todos los elementos disponibles.

Se hace hincapié en integrar Liferay con la base de datos corporativa, y en un futuro, con el Directorio Activo LDAP para obtener los usuarios de la corporación.

Se dispone de dos controladores de dominio (Master-Slave) con Windows que contiene el directorio activo (Active Directory).

Se dispone de un pequeño Data Center (CPD) donde van a estar la mayoría de servidores. Un CPD es un espacio físico especialmente acondicionado para albergar equipos informáticos como ordenadores, equipos de red, sistemas de almacenamiento, etc.

Un CPD puede referirse a un edificio completo con capacidad para 100.000 ordenadores, a un conjunto de edificios, a una gran sala con cientos de ordenadores, una jaula ubicada en una sala o a un pequeño rincón de una habitación. Todos ellos son datacenters.

Se dispone de un servidor de Oracle RAC con alta disponibilidad, un servidor físico a cada CPD conectado mediante fibra óptica para garantizar la disponibilidad del servicio en caso de fallo de un nodo.

También existen sistemas de virtualización de servidores y de clientes, tanto VMware como Citrix. Por tanto, se va a usar estos sistemas para montar el entorno Liferay. Concretamente usaremos la virtualización con VMware, creando una máquina virtual con una versión SUSE Linux Enterprise Server para VMware aprovechando las ventajas multiplataforma que nos garantiza Liferay.

Una vez definido el entorno, se ha de elegir el servidor de aplicaciones o contenedor web de Liferay que se adapte mejor a las necesidades. Después de investigar, Tomcat es el servidor elegido por ser de fácil uso y el más extendido. Como se ve en la figura 6.1, se aprecia su interacción de los componentes mencionados con el Servidor Tomcat y la Base de datos Oracle.

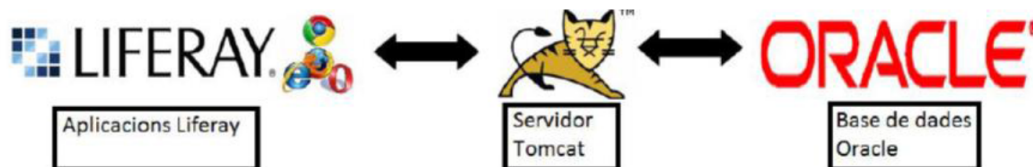


Figura 6.1: Interacción entre tecnologías.

Debido al alcance y versatilidad de Liferay, este sistema se puede ir ampliando y terminar integrándose junto a muchos más sistemas del entorno corporativo. Como, por ejemplo, el gestor documental Alfresco, con nuestro servidor de correo interno, la Nas (almacenamiento conectado en red) de la empresa, etc.

6.1.2. Entorno de desarrollo

Definido ya el entorno en el cual se desplegará la aplicación, toca definir qué entorno de desarrollo se va a usar para crear el portlet de la aplicación. Se ha elegido la herramienta Eclipse por ser uno de los IDEs más populares del mercado, por ser la herramienta usada durante toda la carrera y por la capacidad de integración con otras herramientas, como por ejemplo, SDKs (kit de desarrollo de software). Liferay no es una excepción y la versión 6 incorpora plugin de desarrollo con Eclipse, facilitando el desarrollo de portlets, hooks i ext-plugins ya descritos en la sección anterior.

Para construir el portlet, se necesita usar los plugins SDK que proporciona Liferay y añadirlos a Eclipse, juntamente con la herramienta ANT (Another Neat Tool - herramienta Open-Source utilizada en la compilación y creación de programas Java), la cual realizará las tareas de compilación y acoplamiento. Esta imagen muestra la interacción de Eclipse con Java, Plugins SDK, ANT y los plugins Liferay que se pueden desarrollar.

6.1.3. MVC del portlet a desarrollar

En el apartado de análisis de frameworks se ha tratado los framework mostrando su potencial. En este proyecto se ha escogido usar dos que combinado son potentes: el framework web Spring MVC y el framework de persistencia Hibernate. Es una de las combinaciones más usadas para el desarrollo web de hoy en día.

Será Hibernate el enlace entre los objetos Java y las tablas de la base de datos. De esta forma Hibernate controlara la persistencia de datos. Este framework se complementa con Spring MVC el cual permite inyectar objetos instanciados dentro de otros objetos.

Spring MVC junto a Hibernate también se puede dividir en tres capas:

Modelo

Es la capa encargada de encapsular toda la lógica de negocio de la aplicación y que se puede dividir en dos capas.

- **Lógica de negocio:** Contiene las clases Java para construir la capa de presentación. Por tanto, es la que gestiona las peticiones del controlador para dar una respuesta acorde con la solicitud.
- **Capa de acceso a datos (DAO),** son las clases Java que se encargan de gestionar toda la interconexión con el sistema gestor de base de datos, también contiene un ORM (mapeo objeto-relacional), técnica de programación para convertir los datos de los lenguajes orientados a objetos en su representación en la base de datos relacional. El motor de persistencia Hibernate guarda la información de un objeto de forma permanente. Esta capa solo se comunica con la lógica de negocio.

Vista

Es la página HTML, construida con JSP (JavaServer Pages) y usa la tecnología Javascript. El código provee de datos dinámicos a la página. Por tanto, podemos decir que es la respuesta del controlador y lo que se le presenta al usuario.

Controlador

Es el eje central de la arquitectura. Se encarga de gestionar todas las peticiones, validar los inputs recibidos y dirigir estos a la capa de lógica de negocio. Solo se comunica con la lógica de negocio y responde a través de la vista.

6.1.4. Diseño de la interfaz de usuario

Para la interfaz gráfica, se ha seguido con las directrices marcadas por la responsable de la imagen corporativa. La principal premisa es que el diseño fuese adaptable a los distintos dispositivos. Liferay ofrece tecnologías que cumplen con estos requisitos, facilitando la labor. Otro punto a tener en cuenta, se muestra una imagen neutra y común, siempre visible que se encargará de representar la imagen corporativa independientemente de la sección. Se ha decidido implantar una cabecera común, donde estarán los enlaces a los tres grandes bloques que conforman el portal web, que las sección de Gestiona Vinaros, Gobierna Vinaros y Vive Vinaros. También vamos a añadir herramientas comunes para facilitar la navegación al cliente. Por último, la parte que viene abajo de la cabecera se ha dejado en blanco puesto que su diseño es personalizable por cada responsable de sección , No se implanta ninguna restricción en este apartado.

Propuesta de la interfaz gráfica

Se plantean dos propuestas de interfaz gráfica. En un primer momento, se intenta ofrecer dos propuestas de matices opuestos para que los responsables elijan el camino correcto a seguir.



Figura 6.2: Propuesta gráfica 1.



Figura 6.3: Propuesta gráfica 2.

Como se ve, la primera propuesta 6.2, es más colorida, más viva quizá, aunque no encaja en la idea que quiere ofrecer el ayuntamiento. Si en la estructura y elementos, que como vemos están las tres grandes secciones diferenciadas, y elementos generales, pero no en imagen donde buscan más seriedad y elegancia. Es por ello que se ha seleccionado la segunda propuesta que se ve en la figura 6.3 como vía válida para seguir trabajando.

Parte común del portal - Cabecera



Figura 6.4: Parte común de todo el portal.

La figura muestra la propuesta mejorada. Este es el aspecto que va a tener el portal representando la imagen corporativa. En la figura 6.4 se observa el logo, enlace a las tres grandes secciones, un buscador y la posibilidad de cambiar de idioma. En un futuro, es posible añadir en la derecha el tiempo y información relevante.

Capítulo 7

Implementación

Para implementar el portal se han usado portlets de implementación propia, como el Portlet de opinión, y algunos portlets ya definidos en el portal. El portlet de desarrollo propio es el encargado de gestionar las opiniones enviadas por las entidades aceptadas por el ayuntamiento. Su función es recoger las distintas opiniones en formato de noticia y mostrar estas ordenadas según la fecha de publicación, las más recientes primero. Este portlet cumple con la premisa de ser adaptable a las distintas vistas. También ofrece una sección privada donde solo puede acceder el administrador y en ella se pueden administrar las opiniones.

7.1. Portlet desarrollado - 'Opiniones'

Se ha usado Spring MVC para desarrollar el portlet de opiniones. Cuando se usa Spring portlet en Liferay se puede usar una de las siguientes opciones:

- Spring portlet MVC + Liferay service Builder
- Spring portlet MVC + implementación Spring DAO (Data Access Object) con Hibernate

Actualmente en Liferay existe una herramienta denominada 'Service Builder' que te permite delegar la creación de clases y métodos necesarios para desarrollar la lógica de acceso a los datos de forma automática, definiendo las entidades y el tipo de datos a la base de datos. Pero en este desarrollo no se ha utilizado ya que se pierde el control de esta lógica, porque al ser Liferay el que crea automáticamente los archivos hace menos manejable las interacciones con la base de datos. En cambio, usando la alternativa queda más claro la comunicación entre capas del modelo, vista y controlador que es lo deseado.

Por tanto, se ha seleccionado usar Spring portlet MVC junto con la implementación de Spring DAO con Hibernate. Además, así se aprovecha la ventaja que proporciona Spring con el uso de anotaciones. El portlet se ha desarrollado utilizando esta funcionalidad. Se usa:

- Spring portlet MVC

- Spring portlet DAO con Hibernate
- Anotaciones de Hibernate y de Spring
- Java Persistence API (JPA)

Configuración

El portal Liferay dispone de archivos .jar de los frameworks Spring y Hibernate, con lo cual se puede, o bien utilizar los propios .jars que proporciona Liferay o bien se añaden manualmente dentro del portlet. Se ha optado por añadir los .jar dentro del portlet en su carpeta de librerías.

La figura 7.1 muestra la carpeta LiferaySpring-portlet docroot WEB-INF lib.

Al tener las librerías internamente, no se tienen que añadir dependencias en el archivo de configuración liferay-plugin-package.properties. Así, este archivo de configuración queda como se observa en la figura 7.2.

Una vez configuradas y añadidas las librerías necesarias, se pasa a configurar los archivos XML necesarios para el correcto funcionamiento del portlet.

- *web.xml*: Archivo donde se configuran las propiedades del servidor, juntamente con los posibles recursos de la base de datos. Este fichero determina donde se almacena el fichero de configuración de Spring, en nuestro caso, `'/WEB-INF/portletApplicationContext.xml'`.
- *spring.properties*: archivo de propiedades para hacer la conexión con la base de datos.
- *portletApplicationContext.xml*: archivo que contiene la configuración de Spring de la aplicación y que es referenciado en el archivo *web.xml*. Tiene las siguientes partes:
 - *viewResolver*: gestor de las vistas de la aplicación.
 - *spring.properties*: referencia al archivo antes mencionado con las propiedades de la conexión.
 - *Configuración de Hibernate*: para hacer el mapeo de atributos entre la base de datos relacional y los objetos de la aplicación, se establece que la clase Java donde se ha de realizar el mapeo se ubicará al paquete específico para los objetos del dominio.
 - Referencias a las clases *DAO* y las clases *Services*.
- *portlet.xml*: archivo del portlet propio donde se definen sus características. Se necesita definir un *FrontController*, que será el *DispatcherPortlet*. También se debe mencionar donde se ha definido el fichero *application context*.
- *LiferaySpring-portlet.xml*: Archivo que se referencia en el *portlet.xml* y que contiene los controladores necesarios para configurar la gestión del portlet, para que se puedan invocar por su *URL*.
- *Configuración de idiomas*: archivos para definir las etiquetas de los distintos idiomas de la aplicación. Se pueden crear tantos archivos como idiomas sean necesarios.

- *Language_properties*
- *Language_ca_properties*

Capa de acceso a los datos

- *Domain*: Primero se definen las clases de los objetos que se quieren mapear con *Hibernate* denominadas *domain*. Estas usarán las anotaciones *JPA* de *Hibernate*.
- *DAO*: esta parte se encarga de hacer las interacciones con la base de datos mediante el mapeo del '*domain*' definido anteriormente.

Capa lógica de negocio

En esta capa intermedia se controla la lógica de negocio del portlet propio. Para hacerlo se necesita definir el *service*, el cual realizará la comunicación entre el controlador y el *DAO*. Este también usa las anotaciones del framework *Spring*.

Capa de presentación

- **Controlador**: Se implementa el controlador el cual gestiona las peticiones del portlet. En el controlador cada método es invocado por el *Request Parameter mapping*'. Este también usa las anotaciones del framework *Spring* como *@Autowired* para la inyección de dependencias.
También proporciona anotaciones para crear facilidades de mapeo para peticiones de portlet típicas como son *@RenderMapping* para solicitar paginas y *@ActionMapping* para realizar distintas acciones como añadir, modificar y eliminar objetos.
- **JSP (JavaServer Pages)**: las vistas se han creado mediante archivos JSP. Se han usado las etiquetas de formularios que tienen la capacidad de recuperar los datos de los modelos de objetos. También se ha usado '*portlet URL*' para invocar distintas acciones del controlador mediante etiquetas.
Se ha usado un tema gratuito obtenido de Liferay, el cual ofrece una base con los requisitos mínimos que requiere una plantilla. Sobre este, se ha modificado el estilo *CSS (Cascading Style Sheets)* para adaptarlo a las características esperadas del portal.

7.2. Portlets integrados en Liferay

Se ha aprovechado la potencia que tiene Liferay usando portlets ya implementados y testados por la comunidad.

Se han seleccionado estos portlets para la implementación porque cada uno, en su medida, cumple con el objetivo de hacer el portal más participativo. Gracias al portlet '*Calendar*', los usuarios están informados de todos los detalles en la vida interna de la corporación, y ello,

de forma dinámica y fácil. Con el *'Foro'*, se tiene una herramienta de debate donde los usuarios pueden expresar sus opiniones y los responsables hacer consultas concretas para saber qué piensan los usuarios. Con las *'Encuestas'*, se obtienen respuestas a preguntas concretas, de una forma más estricta y numérica. En caso de tener que decidir, se puede consultar la opinión de ciertos usuarios de esta forma. El *'Formulario de contacto'* ofrece una vía alternativa para la comunicación entre el pueblo y la corporación. Finalmente, se han implementado herramientas comunes, como son el *'Buscador'* y el gestor de *'Idiomas'*, que permiten acceder a todo el contenido y en el idioma que más se adecue a las características del usuario de la aplicación

7.2.1. Buscador

El buscador permite encontrar cualquier activo dentro del portal. Cumple con el requisito de formar parte de la zona común (cabecera) y ofrecer enlace a todo tipo de contenido sin importar su sección.

7.2.2. Idiomas

Ubicado en la cabecera, permite cambiar el idioma del portal. De esta forma, se alcanza la necesidad de ofrecer el contenido en distintos idiomas según el gusto del usuario.

7.2.3. Calendar

Este portlet cumple con muchos de los requisitos descritos por el cliente. Un calendario donde se pueden mostrar diferentes eventos agrupados por categorías. Para ello, se crean calendarios con distintos nombre, que simulan categorías, y se muestran en el portlet *Calendar*. El usuario puede seleccionar los calendarios que desea ver de forma sencilla pulsando en el nombre. De esta forma, este portlet es de gran utilidad en distintas secciones, desde mostrar la agenda de los concejales, hasta mostrar la agenda cultural con actos y eventos. Ofrece filtros y se adapta a la dimensión de la pantalla, por lo que es muy acertado su desarrollo.

7.2.4. Foro

Este ofrece un foro completamente funcional. Gestionable en categorías y permisos para poder mostrarlo en distintas secciones filtrando su contenido. Gracias a ello, se obtiene una herramienta administrable por los distintos responsables de sección. Permite configurar roles y usuarios para fomentar la seguridad y el buen uso del foro.

7.2.5. Encuestas

Permite gestionar una batería de encuestas y mostrarla en cualquier parte y sección del portal. Es sencillo de usar y configurar. Simplemente se crea la pregunta, sus respuestas y los

permisos de los usuarios que pueden responder. Automáticamente se puede ofrecer al usuario para su uso.

7.2.6. Formulario de contacto

Vía de comunicación entre usuarios anónimos y el portal. Con la medidas de seguridad necesarias para evitar el envío de correo masivo, el formulario permite modificar y gestionar hasta el último detalle de campos y formatos.

7.3. Base de datos

Gracias al framework de persistencia Hibernate, no se ha tenido contacto directo con el gestor de base de datos MySql puesto que Hibernate se encarga de ello haciendo que sea transparente. Se han definido las clases con sus atributos, y automáticamente, el framework crea y gestiona la información en la base de datos.

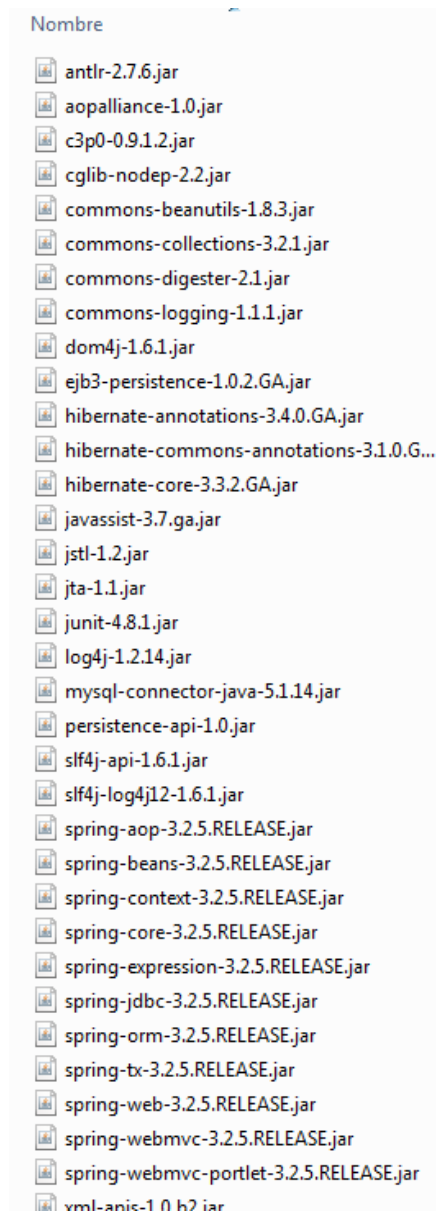
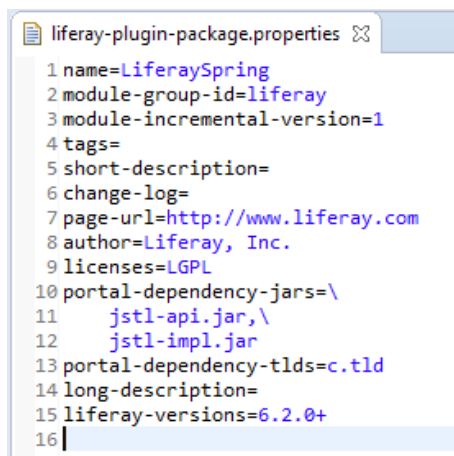


Figura 7.1: Librerías frameworks Spring y Hibernate.



```
liferay-plugin-package.properties
1 name=LiferaySpring
2 module-group-id=liferay
3 module-incremental-version=1
4 tags=
5 short-description=
6 change-log=
7 page-url=http://www.liferay.com
8 author=Liferay, Inc.
9 licenses=LGPL
10 portal-dependency-jars=\
11     jstl-api.jar,\
12     jstl-impl.jar
13 portal-dependency-tlds=c.tld
14 long-description=
15 liferay-versions=6.2.0+
16 |
```

Figura 7.2: Propiedades de configuración.

7.3.1. Ejemplo de la interfaz gráfica implementada

A continuación se observa una serie de imágenes que muestran el comportamiento del portal según los distintos dispositivos de los que se accede al portal. Se recorre desde la versión más grande, la de escritorio, hasta la más pequeña, la de un móvil .

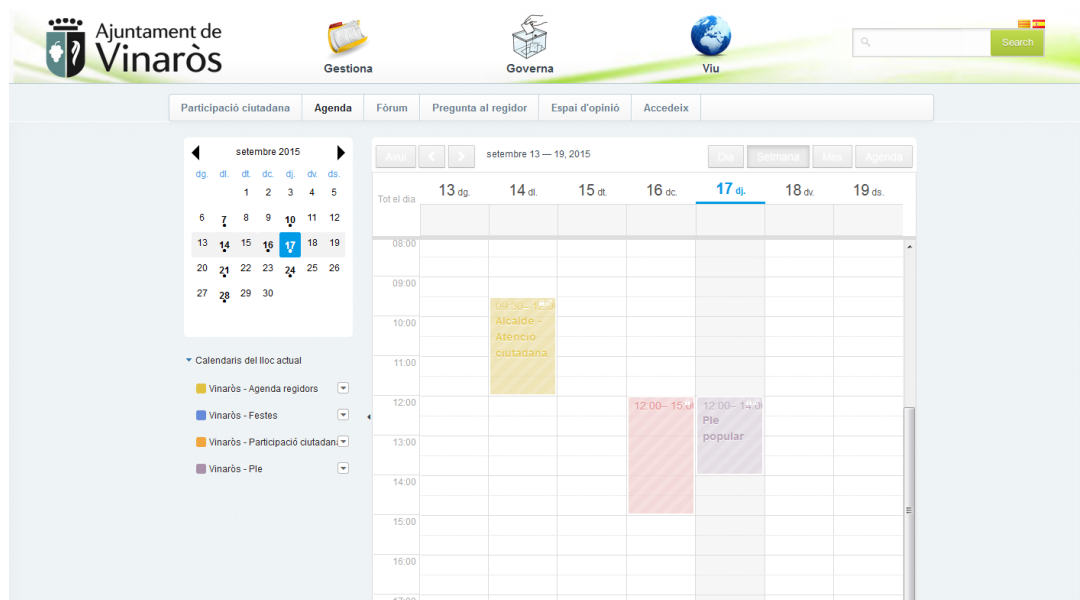


Figura 7.3: Interfaz gráfica a 1280px.

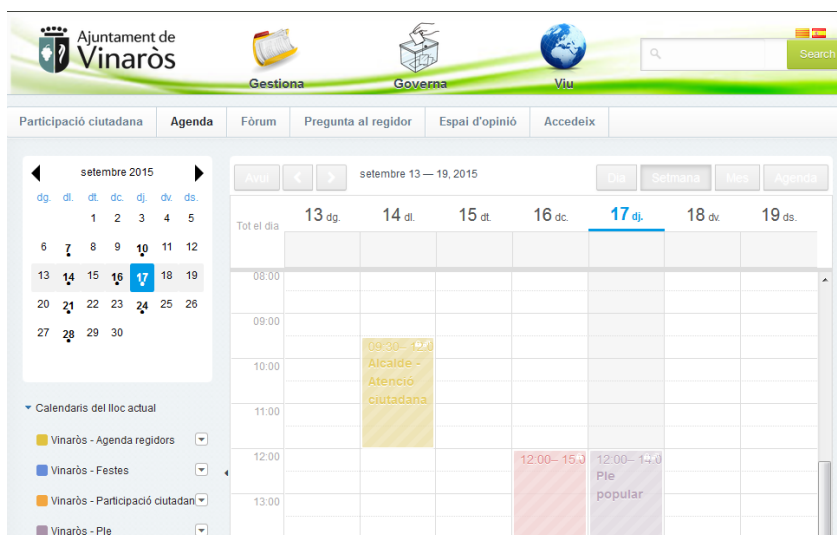


Figura 7.4: Interfaz gráfica a 800px.

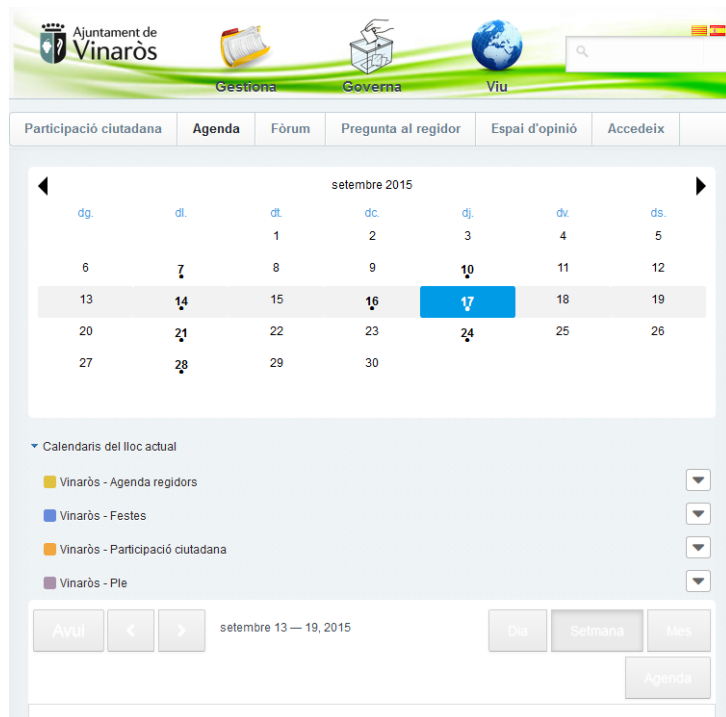


Figura 7.5: Interfaz para tabletas.

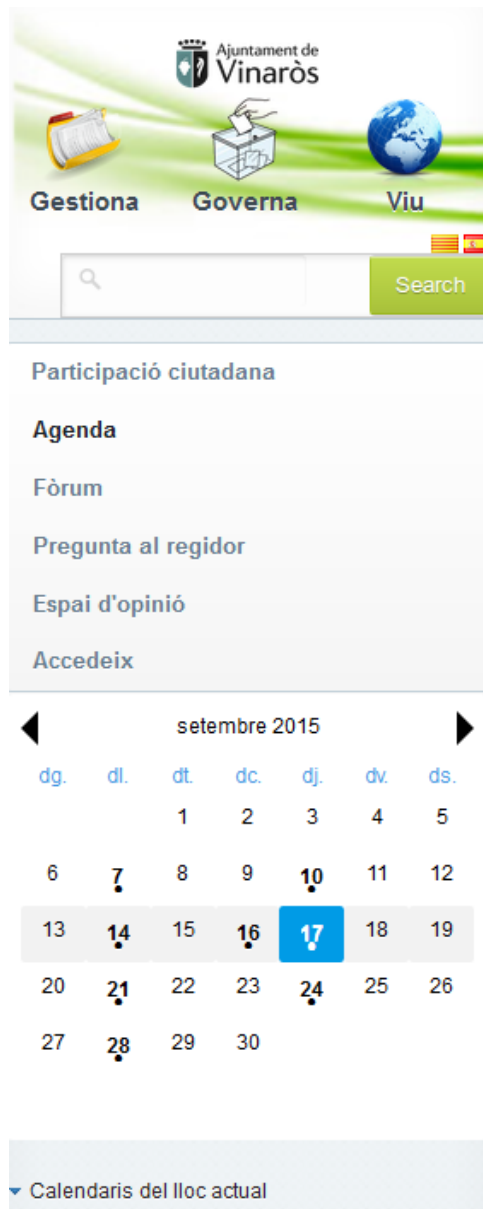


Figura 7.6: Interfaz para móvil.

Capítulo 8

Conclusiones

Este proyecto ha servido para realizar un análisis y diseño completo. El departamento de informática se nutrirá de este para redactar el pliego de condiciones, pliego con el cual negociar la externalización de la implementación con las empresas interesadas. El portal queda definido con un nivel de precisión alto. Para el futuro, sólo queda pulir detalles mínimos que aporten características externas, como sería la interfaz gráfica y sus florituras. Lo que respecta a la parte interna, se han dejado definidos los cimientos de la aplicación de forma que sean sólidos y consistentes, así como, que sobrevivan al paso de los años sin tener que sufrir modificaciones considerables.

El hecho de realizar el portal con la herramienta Liferay, desconocida hasta el momento, ha propiciado la inmersión en el mundo de los gestores de contenido por la parte que me pertoca. Liferay se convierte en una herramienta a tener en cuenta para futuras implementaciones. La experiencia es positiva, y sorprende gratamente que el hecho de que ser Open Source no le penaliza para competir con otras soluciones privadas de empresas como IBM o Oracle.

Tras reuniones, entrevistas, divagaciones, y después, más reuniones, se vislumbra la cantidad de trabajo que hay detrás de cada línea de código. Cada granito de arena es importante a la hora de realizar un proyecto. Más aún, en un proyecto de la envergadura del citado en este documento. La cantidad de gente y recursos a coordinar dejan entrever la parte oculta en la realización de proyectos. No se ha presentado sencillo, aunque, finalmente se ha conseguido sortear la mayoría de obstáculos para obtener un resultado satisfactorio.

En líneas generales, el proyecto ha servido para asentar conocimientos, y a la vez, obtener nuevos de ellos fruto de la experiencia. Durante el transcurso de este, se ha recorrido un amplio abanico del temario de los estudios. Con nostalgia se recuerda los días de prácticas y ejercicios donde se simulaba la vida laboral en el centro de estudios. Aunque, en algún que otro día, se cuestionó la verdadera necesidad de adquirir ciertos conocimientos, que para sorpresa, han sido indispensables para la finalización de este proyecto. En este sentido, la labor del profesorado pasa inadvertida ante los ojos del estudiante.

Después de vivir la experiencia laboral, se han desechado mitos obtenidos durante los estudios. La teoría nada tiene que ver con la práctica. A la hora de la verdad, la experiencia es un

grado que prima con solvencia al quien la tiene.

En cuanto al lugar de la estancia, la calidad humana del grupo es inmejorable. Han sido un pilar desde el principio, atendiendo y resolviendo pacientemente las dudas surgidas. No solo en el ámbito estrictamente profesional, sino también, en el aspecto más sociable. La empresa se convirtió en un lugar agradable donde trabajar, compartir experiencias y anécdotas. El hecho de respirar este tipo de atmósfera facilita la inserción, la motivación y la productividad.

Estas palabras simbolizan la consecución de los estudios superiores. Con ellas se cierra una etapa maravillosa en la que se han acumulado experiencias, amistades, historias y conocimientos. Se sale de ella preparado para afrontar la vida laboral, y como no, seguir forjando día tras día mas conocimientos y experiencias.

Bibliografía

- [1] Liferay Portal 6.2 Developer's Guide
<http://www.liferay.com/es/documentation/liferay-portal/6.2/development>
- [2] Using Liferay Portal 6.2
<http://www.liferay.com/es/documentation/liferay-portal/6.2/user-guide>
- [3] Portlet Development - Development — Liferay. Available at
<https://www.liferay.com/es/documentation/liferay-portal/6.0/development/-/ai/portlet-development>
- [4] The Plugins SDK - Development — Liferay. Available at
<http://www.liferay.com/es/documentation/liferay-portal/6.1/development/-/ai/the-plugins-s-3>
- [5] Sezov, R. (2012). Liferay in action
- [6] Java Community Process. JSR-000286 Portlet Specification, (Final Release)
<http://jcp.org/aboutJava/communityprocess/final/jsr168/index.html>
- [7] Java Community Process. JSR-000168 Portlet Specification, (Final Release)
<http://jcp.org/aboutJava/communityprocess/final/jsr286/index.html>
- [8] Bauer, C., & King, G. (2005). Hibernate in action
- [9] Hibernate ORM - Hibernate ORM
<http://hibernate.org/orm/>
- [10] Walls, C., & Breidenbach, R. (2005). Spring in action. Dreamtech Press
- [11] Spring
<http://spring.io/>