Universitat Oberta de Catalunya

Màster Universitari en Programari Lliure

PROJECTE FINAL DE MÀSTER

PORTAL LIFERAY

SOBRE INCLUSIÓ SOCIAL DE JOVES A TRAVÉS DE L'OCUPACIÓ

Autor: César Mones Valanzuela

Tutor UOC: Gregorio Robles Martínez
Tutor extern: Jordi Tolosà Bel

Juny 2016

Llicència de publicació del document

Índex de continguts

Índex de figures

1 Introducció

El projecte consisteix en desenvolupar un portal web sobre inclusió social de joves a través de l'ocupació. El portal inclourà informació sobre el projecte, normativa europea, base de dades de bones pràctiques, *links* a eines gratuïtes sobre ocupabilitat, accés a xarxes socials del projecte, etc. Es tracta de satisfer una necessitat reial que s'ha plantejat recentment al Departament de Benestar Social de l'Ajuntament.

En primer lloc s'hauran d'establir els requeriments de l'aplicació i, posteriorment, es realitzà l'anàlisi i disseny de l'aplicació. Per tal de dur a terme el desenvolupament s'haurà de preparar l'entorn de treball.

L'aplicació estarà implementada amb el CMS Liferay, amb el que es crearan els *portlets* necessaris per a dotar de les funcionalitats requerides a l'aplicació. El motor de base de dades serà MySQL.

2 Pla de treball

El pla de treball inclou un calendari de desenvolupament i la planificació temporal del projecte.

2.1 Calendari de desenvolupament

El Calendari de desenvolupament mostra les diferents fites del projecte:

PAC1 (11/3/2016)

La primera fita es correspon amb el lliurament d'aquest document. S'haurà de lliurar la següent documentació:

- Descripció del projecte a realitzar
- Descripció dels objectius i les tecnologies a emprar
- Definició d'un pla de treball amb el calendari de desenvolupament i la planificació temporal

PAC2 (9/4/2016)

Aquesta fita marca la finalització de l'anàlisi i disseny del projecte. S'haurà de lliurar la següent documentació:

- Anàlisi funcional del portal
- Anàlisi de l'arquitectura Liferay
- Disseny del portal: diagrames UML (diagrama de casos d'ús, diagrama de classes, etc.)

PAC3 (14/5/2016)

Aquesta última PAC consisteix a dur a terme la fase d'implementació del projecte. S'hauran de realitzar i documentar les següents tasques:

- Instal·lació de l'entorn de treball
- Desenvolupament de *portlets* propis
- Desenvolupament del portal
- Testatge i correcció d'errors

Memòria (11/6/2016)

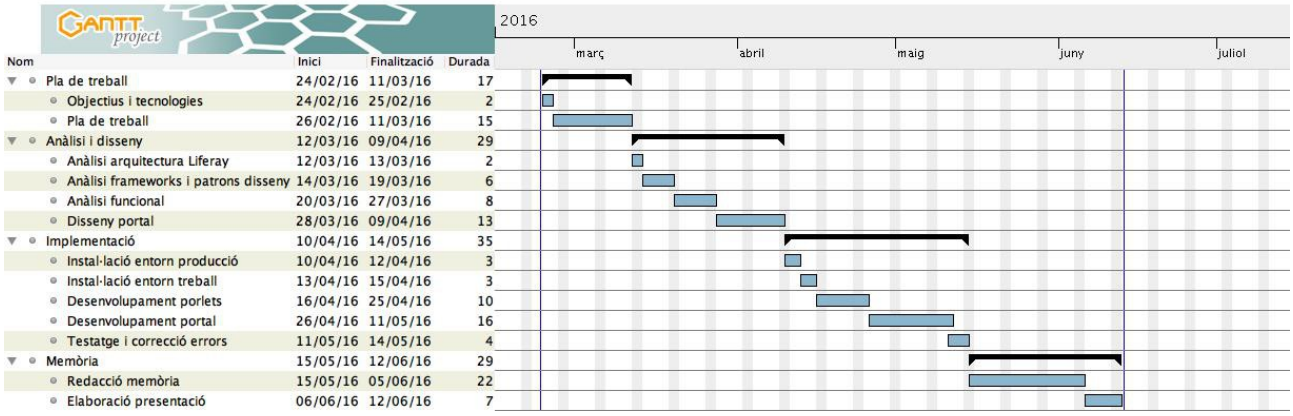
L'última fita es correspon amb el lliurament de la memòria:

- Redacció de la memòria final del projecte

- Elaboració d'una presentació del projecte

2.2 Planificació temporal

El següent diagrama de Gantt mostra la planificació temporal del projecte amb les fites previstes:



Nom	Inici	Finalització	Durada
Pla de treball	24/02/16	11/03/16	17
Objectius i tecnologies	24/02/16	25/02/16	2
Pla de treball	26/02/16	11/03/16	15
Anàlisi i disseny	12/03/16	09/04/16	29
Anàlisi arquitectura Liferay	12/03/16	13/03/16	2
Anàlisi frameworks i patrons disseny	14/03/16	19/03/16	6
Anàlisi funcional	20/03/16	27/03/16	8
Disseny portal	28/03/16	09/04/16	13
Implementació	10/04/16	14/05/16	35
Instal·lació entorn producció	10/04/16	12/04/16	3
Instal·lació entorn treball	13/04/16	15/04/16	3
Desenvolupament porlets	16/04/16	25/04/16	10
Desenvolupament portal	26/04/16	11/05/16	16
Testatge i correcció errors	11/05/16	14/05/16	4
Memòria	15/05/16	12/06/16	29
Redacció memòria	15/05/16	05/06/16	22
Elaboració presentació	06/06/16	12/06/16	7

Figura 1. Diagrama de Gantt.

3 Anàlisi de l'aplicació

3.1 Requisits funcionals

El projecte consisteix a desenvolupar un portal web sobre inclusió social de joves a través de l'ocupació.

El portal ha de tenir un menú principal amb els següents apartats:

- Informació sobre el projecte i els organismes que hi intervenen.
- Informació sobre normativa europea.
- Base de dades de bones pràctiques en aquesta matèria.
- Recull d'eines gratuïtes sobre ocupabilitat.
- Fòrums on es puga debatre sobre aquesta matèria.
- Recull de recursos amb oferta de treball per a joves disponibles en Internet.

A més, el portal ha d'incorporar els elements següents:

- Blog de notícies.
- Calendari d'esdeveniments.
- Xarxes socials.

Per altra banda, el portal web haurà d'ésser multilingüe.

El portal tindrà una part privada i una pública:

- La part privada s'utilitzarà per a l'administració del portal i la provisió de continguts.
- La part pública estarà destinada al públic en general i, especialment, als col·lectius a què va dirigida el portal: professionals que treballen en el àmbit de la inclusió social de joves i joves que cerquen ocupació.

Per tant, els usuaris del portal tindran algun dels següents rols:

- Administrador: usuaris amb permís d'administració, configuració i definició d'usuaris del portal.
- Usuari amb privilegis: usuaris amb permís de provisió de continguts als diferents elements del portal (pàgines, bones pràctiques, blog, fòrums, calendari).
- Usuari: usuaris amb permís per a intervindre en els fòrums.
- Invitat: usuaris que no s'identifiquen i, per tant, només tenen permís per a visualitzar els continguts públics del portal.

3.2 Diagrama de casos d'ús

Els casos d'ús són la descripció dels passos o activitats que s'han de realitzar per dur a terme algun procés. A continuació es mostren els casos d'ús del portal en base als requisits funcionals que hem definit i als actors o rols que intervenen.

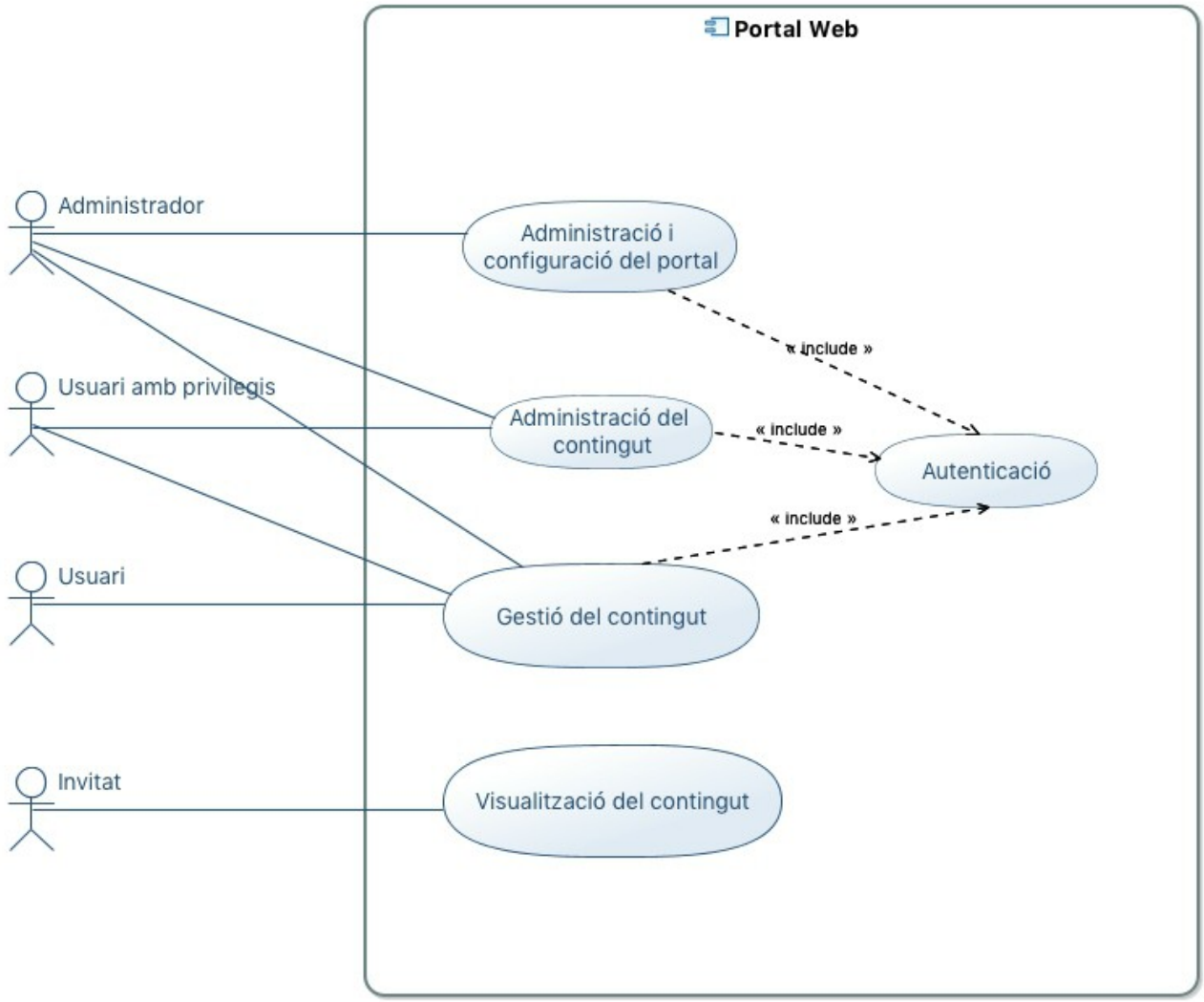


Figura 2. Diagrama de casos d'ús. Accés al portal.

Cas d'ús 1: Autenticació d'usuaris

Descripció	Permet als usuaris identificar-se en el sistema. Una vegada identificats, els usuaris obtenen permisos sobre els diferents elements del portal en funció del seu perfil
Actors implicats	Administrador, Usuari amb privilegis i Usuari
Casos d'ús relacionats	

Precondició	Estar donat d'alta al sistema
Postcondició	Accedir a la part privada del portal
Alternatives de procés i excepcions	L'usuari no pot autenticar-se fins que l'administrador no l'haja donat d'alta

Cas d'ús 2: Administració i configuració del portal

Descripció	Permet administrar i configurar completament el portal i definir els usuaris que hi tenen accés
Actors implicats	Administrador
Casos d'ús relacionats	Autenticació
Precondició	Estar identificat correctament al sistema
Postcondició	Administrar el portal
Alternatives de procés i excepcions	Ha d'existir un portal

Cas d'ús 3: Administració del contingut

Descripció	Permet administrar les pàgines i aplicacions del portal
Actors implicats	Administrador i Usuari amb privilegis
Casos d'ús relacionats	Autenticació
Precondició	Estar identificat correctament al sistema
Postcondició	Administrar les pàgines i aplicacions del portal
Alternatives de procés i excepcions	Els Usuaris amb privilegis no podran administrar el contingut del portal fins que l'administrador no els haja concedit els permisos pertinents

Cas d'ús 4: Gestió del contingut

Descripció	Permet gestionar el contingut del portal
Actors implicats	Administrador i Usuari amb privilegis
Casos d'ús relacionats	Autenticació
Precondició	Estar identificat correctament al sistema
Postcondició	Gestionar el contingut del portal
Alternatives de procés	Els Usuaris amb privilegis no podran gestionar el contingut del

i excepcions	portal fins que l'administrador no els haja concedit els permisos pertinents

Cas d'ús 5: Visualització del contingut

Descripció	Permet visualitzar els continguts del portal
Actors implicats	Administrador, Usuari amb privilegis, Usuari i Invitat
Casos d'ús relacionats	
Precondició	
Postcondició	Visualitzar els continguts del portal
Alternatives de procés i excepcions	

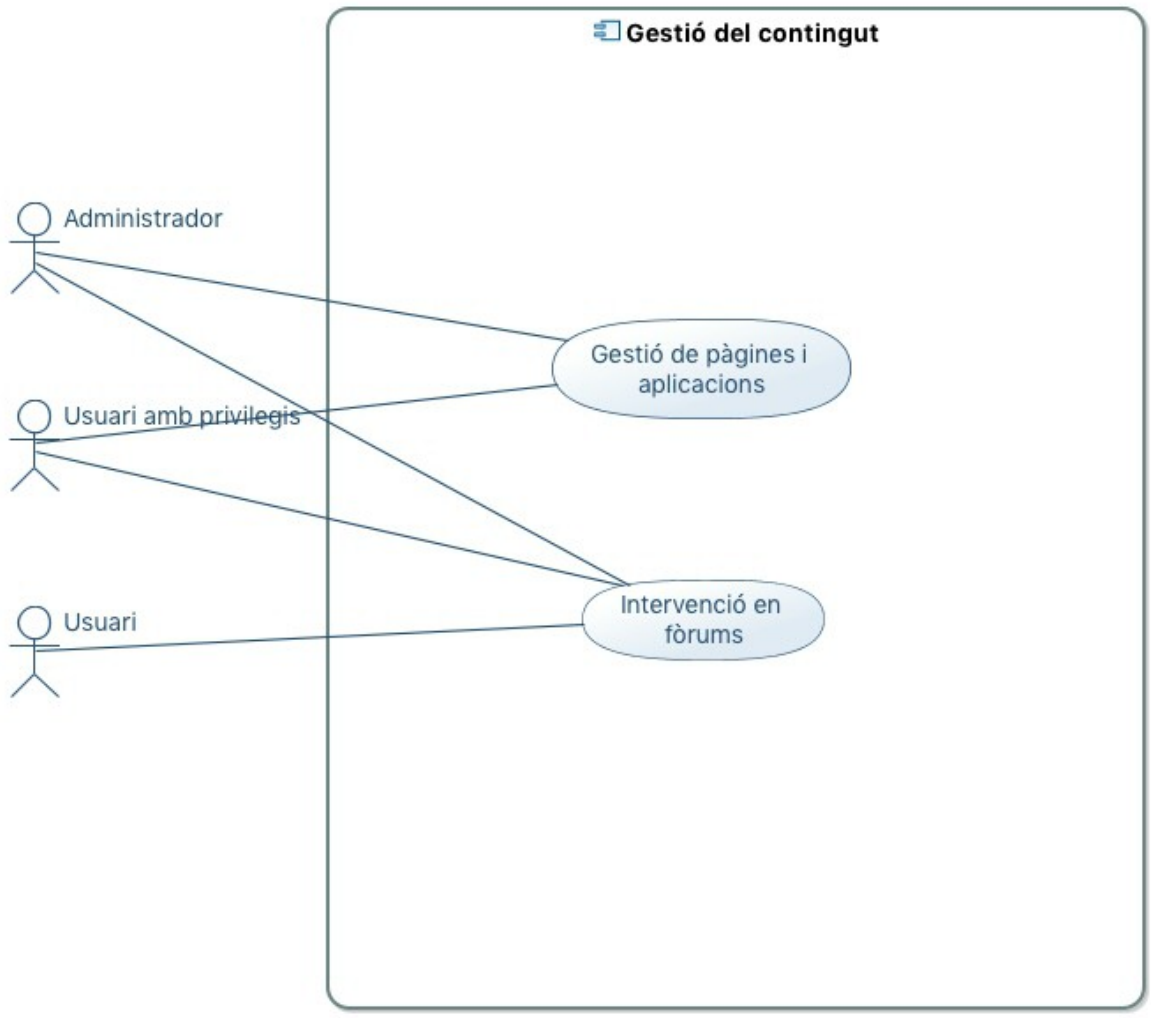


Figura 3. Diagrama de casos d'us. Gestió del contingut.

Cas d'ús 5: Gestió de pàgines i aplicacions

Descripció	Permet gestionar les pàgines i aplicacions (bones pràctiques, blog, fòrums, calendari) del portal
Actors implicats	Administrador i Usuari amb privilegis
Casos d'ús relacionats	
Precondició	Estar identificat correctament al sistema
Postcondició	Gestionar les pàgines i aplicacions del portal
Alternatives de procés i excepcions	

Cas d'ús 6: Intervenció en fòrums

Descripció	Permet intervindre en els fòrums del portal
Actors implicats	Administrador, Usuari amb privilegis i Usuari
Casos d'ús relacionats	
Precondició	Estar identificat correctament al sistema
Postcondició	Intervindre en els fòrums del portal
Alternatives de procés i excepcions	

4 Disseny de l'aplicació

Per tal de realitzar el disseny de l'aplicació és necessari d'aprofundir en les eines i llenguatges de programació concretes que s'utilitzaran en el desenvolupament del projecte.
És per això que en primer lloc analitzaré l'arquitectura del portal Liferay i les possibilitats que ofereix aquesta plataforma per a construir el portal.
Posteriorment explicaré quin serà l'arquitectura del portal i quins components de Liferay utilitzaré per a implementar-lo.
Finalment detallaré el disseny dels *portlets* que s'hauran de desenvolupar.

4.1 Arquitectura Liferay

Liferay és un portal de gestió de continguts de codi obert escrit en Java que té com a principals característiques:

- Desenvolupament de funcionalitats a través de *portlets*.
- Incorpora més de 60 *portlets* integrats en el nucli, que faciliten el desenvolupament de portals web.
- Hibernate per a l'accés a bases de dades (DB2, HSQLDB, MySQL, Oracle, PostgreSQL, etc.)
- Struts, JSF, Facelets, ICEFaces, etc.
- Integració amb LDAP.
- Servidors d'aplicacions JBoss, Tomcat, OC4J, Geronimo, Glassfish, Weblogic, Websphere, Jetty.
- Lucene com a motor d'indexació i cerca de continguts.
- Gestió d'usuaris i permisos.
- Clustering i caches distribuïdes (Ehcache)
- Workflows (Implementació del motor JBoss' jBPM)
- Single Sign on per a l'autenticació i autorització única front als diferents sistemes.
- Personalització de plantilles, CSS i Javascript.
- Basat en estàndards: JSR 127 (JSF), JSR 168 (Porlet Specification), JSR 286 (Porlet 2.0 Specification), JSR 170 (Content Repository), JSR 208 (Java Business

Integration), AJAX, Spring, Struts, Tiles, Velocity, WSRP.

En la següent imatge es mostra l'arquitectura lògica de Liferay:

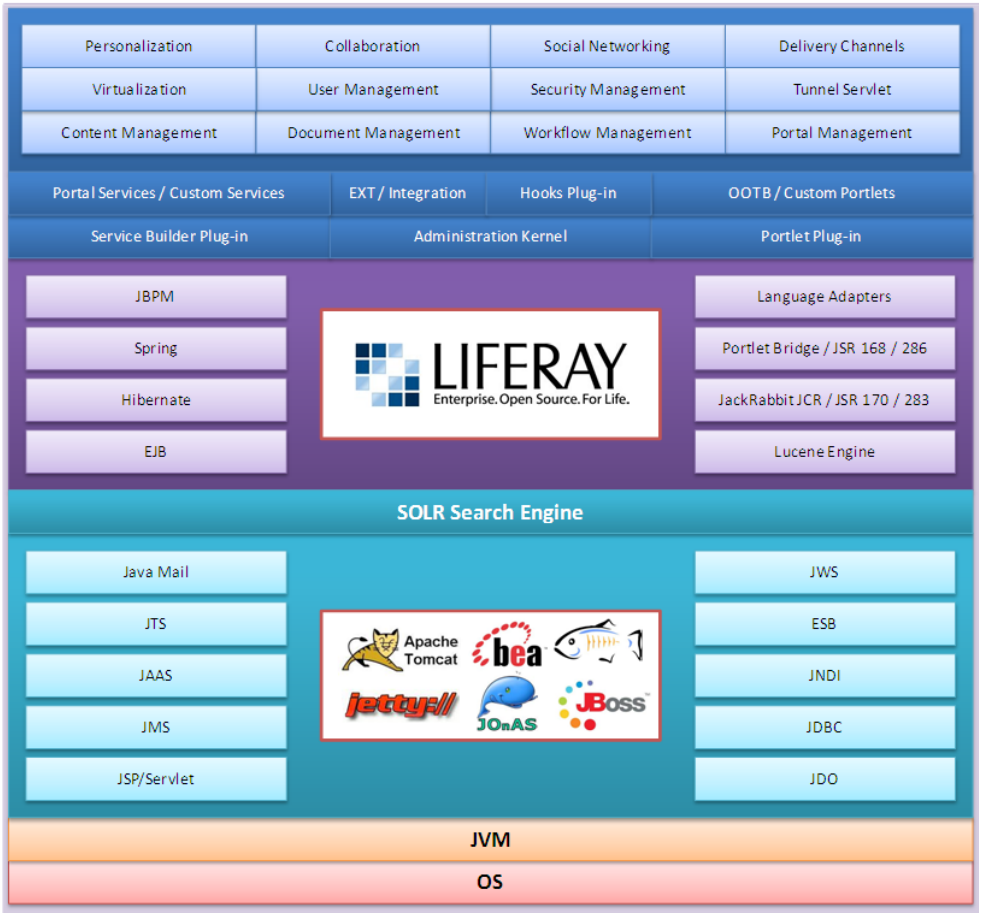


Figura 4. Arquitectura lògica de Liferay.

Liferay ofereix un gran nivell de personalització que permet des de crear noves aplicacions fins a canviar completament l'aspecte del portal. Per a portar-ho a terme, Liferay ofereix les següents eines o *plugins*:

- ***Portlets***

Els *portlets* són aplicacions contingudes dins d'un portal i s'encarreguen de generar continguts dinàmics per a ell.

Existeixen dues especificacions (JSR 168 i JSR 286) que defineixen una sèrie de funcions i classes que han d'implementar les aplicacions per a poder seguir l'estàndard i d'aquesta forma, ser portables d'un portal a un altre.

A nivell tècnic, un *portlet* comença per una classe Java que implementa la classe interfície *Javax.portlet.Portlet*. Els *portlets* només tenen sentit quan estan continguts dins d'un contenidor de *portlets* que sol ser el portal, i interactuen amb els usuaris web amb un paradigma de peticions i respostes (*request/response*).

A diferència dels *servlets* (petits programes que s'executen en el context d'un navegador web), els *portlets* no poden ser cridats de forma directa, és a dir, d'un *servlet* es pot escriure la seua adreça web però d'un *portlet* només es podrà obtenir l'adreça web de la pàgina que ho conté, però no una petició directa sobre ell. Una altra diferència és que els *servlets* poden generar pàgines o documents complets, mentre els *portlets* solament generen una part del contingut.

En Liferay, totes les aplicacions com a wiki, blogs, continguts, etc. són *portlets*, ja que Liferay és un contenidor de *portlets* i si una aplicació vol integrar-se en aquest portal ha de complir amb aquest requisit. A més de les funcionalitats existents, es podrem desenvolupar nous *portlets* a mida per a realitzar altres tasques addicionals que no implementa Liferay per defecte. En el nostre cas, serà necessari implementar dos *portlets* per l'administració i la publicació en el nostre portal de bones pràctiques en l'àmbit de la inclusió de joves a través del treball.

- ***Hooks***

Aquest terme és propi de Liferay. Els *hooks* són una de les maneres que té Liferay de sobreescriure funcionalitats del portal. D'aquesta forma s'evita haver de recompilar el codi del portal cada vegada que es desitja realitzar una modificació en determinats paràmetres interns del portal.

Un clar exemple d'això són les traduccions o els paràmetres de les pàgines. En el cas que es vulga fer algun canvi en la traducció del portal, la via tradicional suposaria modificar el fitxer dels idiomes del portal i recompilar-lo. Amb els *hooks* de Liferay només és necessari generar un *hook*, modificar el fitxer que es desitja i carregar el *hook* al portal. Un altre cas similar succeeix si es vol afegir un paràmetre nou a les pàgines, per exemple perquè no mostren la capçalera.

Existeixen quatre tipus de *hooks* principals:

  - Custom JSPs: Permeten modificar el codi d'un o diversos *portlets* mitjançant la modificació dels seus JSPs.
  - Portal properties: Permeten modificar les propietats del portal o definir-ne de

noves.

- Services: Permeten modificar els serveis del portal directament o afegir-ne de nous.
- Language properties: Permeten declarar noves variables d'idiomes per facilitar la internacionalització del portal. Aquestes variables després seran usades en diferents punts del portal com, per exemple, per a traduir algun text d'un Tema d'Aparença o d'un *portlet*.

Els *hooks* utilitzen una gran varietat de tecnologies (Java, JSP, HTML, XML) ja que depenen de la part del portal que s'ha de modificar.

- ***Layouts***

El *layout* s'encarrega de la disposició dels *portlet* dins de la pàgina, organitzant-los en files i columnes segons la combinació que es dissenye. Cada pàgina pot tenir un únic *layout*, però diverses pàgines poden utilitzar el mateix. Una vegada escollit el *layout* es podran introduir els *portlets* que han d'aparèixer, en les files i columnes escollides de la pàgina.
És important planificar els *layout* i el *theme* de forma conjunta atés que l'estructura dissenyada en el *layout* serà maquetada pel *theme* és a dir, els colors, separacions i amples seran fixats pel *theme* i no pel *layout*. D'aquesta forma, un *layout* pot ser reutilitzat per diversos *themes* i un mateix *theme* pot emprar diversos *layouts*.
Els *layouts* es desenvolupen amb HTML i Velocity.

- ***Themes***

El *theme* és el mòdul de Liferay que s'encarrega de gestionar l'aparença del portal web. Es desenvolupa majoritàriament amb tres tecnologies: CSS, Velocity, Javascript amb Jquery.
És recomanable que els nous *themes* agafen com a base un ja existent o utilitzen les bases que ofereix la SDK.
Dins del mòdul *theme*, els fitxers es troben organitzats en carpetes, agrupats segons la tecnologia que contenen. Entre aquestes carpetes es troba la denominada *_diffs*. En aquesta, s'ha de replicar l'estructura anterior i modificar el fitxer que es vol adaptar, ja que Liferay, al compilar el mòdul de tema, consulta la carpeta *_diffs* i afig els canvis que hi troba a la resta de carpetes. En cas de realitzar canvis en altres carpetes i no en *_diffs*, és possible que, en compilar els canvis introduïts, siguen eliminats.

4.2 Arquitectura del portal

4.2.1 Components del portal

Com hem dit abans, Liferay ofereix una gran quantitat de funcionalitats mitjançant *portlets* ja incorporats en el nucli que simplifiquen la construcció de portals web.
A continuació es detalla quins *portlets* s'utilitzaran per a la implementació dels principals elements del portal identificats en la fase d'anàlisi de requisits:

- Informació sobre el projecte i els organismes que hi intervenen.
  S'utilitzarà el *portlet Web Content* que permet crear, actualitzar i publicar pàgines web.

- Informació sobre normativa europea.
  S'utilitzarà també el *portlet Web Content*.

- Base de dades de bones pràctiques en aquesta matèria.
  Es desenvoluparan dos *portlets* nous:
  - Administració: aquest *portlet* s'integrarà en el panell de control de Liferay i constarà de formularis amb tots els camps de la base de dades per a registrar, modificar, editar i eliminar organitzacions i bones pràctiques.
  - Visualització: aquest *portlet* apareixerà en la part pública del portal i mostrarà les bones pràctiques que s'han introduït al sistema.

- Recull d'eines gratuïtes sobre ocupabilitat.
  S'utilitzarà el *portlet Web Content*.

- Fòrums on es puga debatre sobre aquesta matèria.
  S'utilitzarà el *portlet Message Boards* que és una solució completa de fòrums amb funcions específiques com categories, capacitat de RSS, avatars, arxius adjunts, avenços, llista dinàmica dels últims llocs, estadístiques, etc.

- Recull de recursos amb oferta de treball per a joves disponibles en Internet.

S'utilitzarà el *portlet Web Content*.

- Blog de notícies.
  S'utilitzarà el *portlet Blogs* que inclou la capacitat completa d'edició WYSIWYG i data de publicació, suport de RSS, comentaris relacionats, etiquetes, vincles de marcadors socials, notificacions per correu electrònic de respostes al blog i un sistema de classificació d'entrada.

- Calendari d'esdeveniments.
  S'utilitzarà el *portlet Calendar*, que permet permet als usuaris crear, administrar i buscar esdeveniments. Els esdeveniments poden ser compartits a través de les comunitats, i els recordatoris d'esdeveniments es poden configurar per alertar els usuaris dels propers esdeveniments per correu electrònic, missatgeria instantània o SMS.

4.2.2 Disseny de la interfície del portal

En aquest apartat anem a definir l'estructura de la interfície del portal. En la fase d'implementació s'haurà d'analitzar si és possible adaptar algun dels *layouts* i *themes* disponibles en Liferay a les nostres necessitats o si serà necessari desenvolupar-ne de nous.

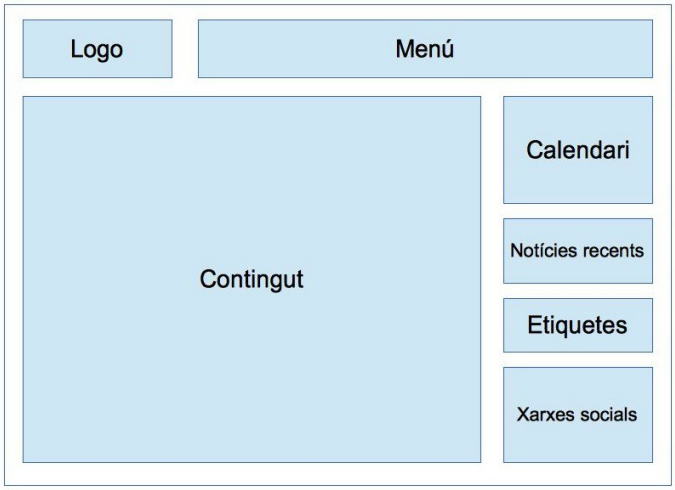


Figura 5. Disseny de la interfície del portal

4.2.3 Disseny dels *portlets*

Com s'ha dit abans, s'han de desenvolupar dos *portlets* amb el quals s'emmagatzemarà en base de dades i es mostrarà en el portal la informació sobre les bones pràctiques en matèria de foment de l'ocupació per a joves en exclusió social.
A continuació exposaré el *framework* que utilitzaré per al seu desenvolupament i detallaré les entitats que hi formen part.

4.2.3.1 Patró MVC a utilitzar

Liferay suporta multitud de *frameworks* per al desenvolupament dels seus *portlets*, incloent Struts, Spring MVC, JSF, i Vaadin. Tanmateix, en aquest projecte utilitzaré Liferay MVC, un *framework* específic de Liferay que, segons els seus creadors, és més senzill, lleuger i fàcil d'entendre que la resta d'opcions. La raó d'aquesta elecció és que un dels objectius del projecte és explorar el màxim possible tota la potencialitat de Liferay per al desenvolupament de portals web. A més, crec que és important familiaritzar-se amb Liferay MVC, ja que és amb aquest *framework* amb què estan desenvolupats tots els *portlets* que Liferay incorpora per defecte.

El *framework* Liferay MVC, com el seu nom indica, està basat en el patró d'arquitectura de programari Model-Vista-Controlador. A continuació s'explica com Liferay MVC implementa els diferents components del patró de disseny MVC:

- Model: la capa de model o de lògica de negoci de l'aplicació conté les dades de l'aplicació i les regles de negoci per a la manipulació d'aquestes dades. Liferay utilitza l'eina Service Builder pera a generar una capa de servei que proporciona una separació entre el model dels objectes i el codi de la base de dades que hi ha per baix. El Service Builder parteix d'un fitxer XML amb la descripció de l'objecte i crea automàticament les capes de model, servei i persistència de forma que el desenvolupador no ha de preocupar-se d'implementar les funcions de creació, lectura, modificació, eliminació i cerca en base de dades.
- Vista: la capa de vista de l'aplicació conté tota la lògica per a la visualització de la les dades per part de l'usuari. Aquesta capa proporciona funcions de manipulació de camps, caselles de verificació i altres elements de formulari, funcions d'ocultació i visualització de dades, etc. Se n'ocupa d'enviar esdeveniments al controlador i

mostrar la resposta del controlador en forma de pàgina HTML.

Liferay utilitza JSP, tecnologies Javascript com ara Jquery i altres eines per a la implementació d'aquesta capa.

- Controlador: el controlador és la capa central d'aquesta arquitectura. Gestiona el flux d'informació entre les capes de model i vista. El controlador, per exemple, és el responsable de determinar quina acció s'ha de realitzar quan un usuari fa un clic en un botó i de llançar l'execució de la funció corresponent que modificarà el model.

4.2.3.2 Diagrama de classes

En els *portlets* d'administració i visualització e Bones Pràctiques només intervenen dos entitats:

- Organització: es tracta d'organitzacions que es distingeixen per les seues bones pràctiques en l'àmbit de la inclusió social de joves a través de l'ocupació.
- Bona pràctica: són les bones pràctiques que porten a terme les organitzacions.

Les organitzacions que es registren en la base de dades podran tenir associades zero, una o vàries bones pràctiques per la qual cosa s'ha establert una relació 1 a * entre les organitzacions i les bones pràctiques.

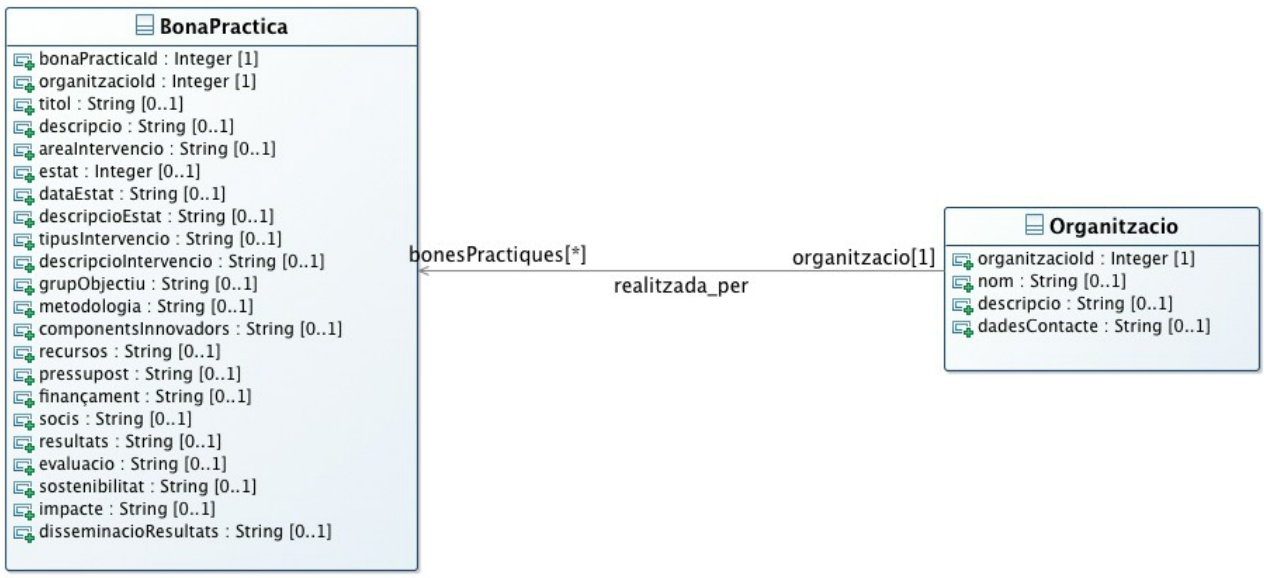


Figura 6. Diagrama de classes.

5 Implementació del portal web

5.1 Instal·lació de l'entorn de desenvolupament

Per a portar a terme la implementació del projecte, el primer que faig és instal·lar l'entorn de desenvolupament, format pels següents elements:

- Java SDK 1.7
- MySQL Community Sever 5.7.9
- Liferay Community Edition GA1 6.2 bundled with Tomcat 7
- Apache Ant 1.9.2
- Liferay Plugins SDK 6.2
- Eclipse Java EE IDE for Web Developers 4.4.0
- Liferay IDE 2.2

Com es pot veure he escollit Eclipse com entorn de desenvolupament. Eclipse és un dels IDE més populars en programació Java, està basat en programari lliure i, mitjançant la instal·lació de l'extensió Liferay IDE, permet el desenvolupament dels diferents tipus de *plugins* de Liferay: *portlets*, *hooks*, *layout templates*, *themes* i *EXT*.
Un cop realitzada la instal·lació del tots els components necessaris, faig les últimes configuracions en Eclipse per a poder iniciar el desenvolupament:

- Registro el servidor Liferay/Tomcat que prèviament he instal·lat per a poder-lo arrancar i parar des d'Eclipse.
- Registro el Liferay Plugin SDK que prèviament he instal·lat per a poder crear nous projectes, importar projectes existents i compilar projectes (mitjançant Apache Ant) sense necessitat de sortir d'Eclipse.

5.2 Desenvolupament dels *portlets*

Finalitzada la instal·lació de l'entorn de desenvolupament, ja puc començar a desenvolupar els *portlets* d'administració i visualització de Bones Pràctiques.
Com he comentat en la fase de disseny, per al desenvolupament dels nostres *portlets* utilitzarem el *framework* Liferay MVC, basat en el patró d'arquitectura de programari Model-Vista-Controlador. Per tal de crear els nostres *portlets* seguirem els següents

passos:

- Creació i configuració d'un projecte de desenvolupament de *portlets*.
- Creació de la capa de model o de lògica de negoci de l'aplicació. Utilitzarem l'eina *Service Builder* de Liferay per a generar automàticament les capes de model, servei i persistència, alliberant-nos d'implementar les funcions de creació, lectura, modificació, eliminació i cerca en base de dades.
- Creació de la capa de vista de l'aplicació. Utilitzarem JSP, Javascript, CSS i el *framework* de construcció d'interfícies AlloyUI per a construir la capa de visualització dels nostres *portlets*.

5.2.1 Creació i configuració d'un projecte de desenvolupament de *portlets*

En primer lloc creo un nou projecte per mitjà de l'assistent d'Eclipse per a la creació de projectes de *plugins* de Liferay:

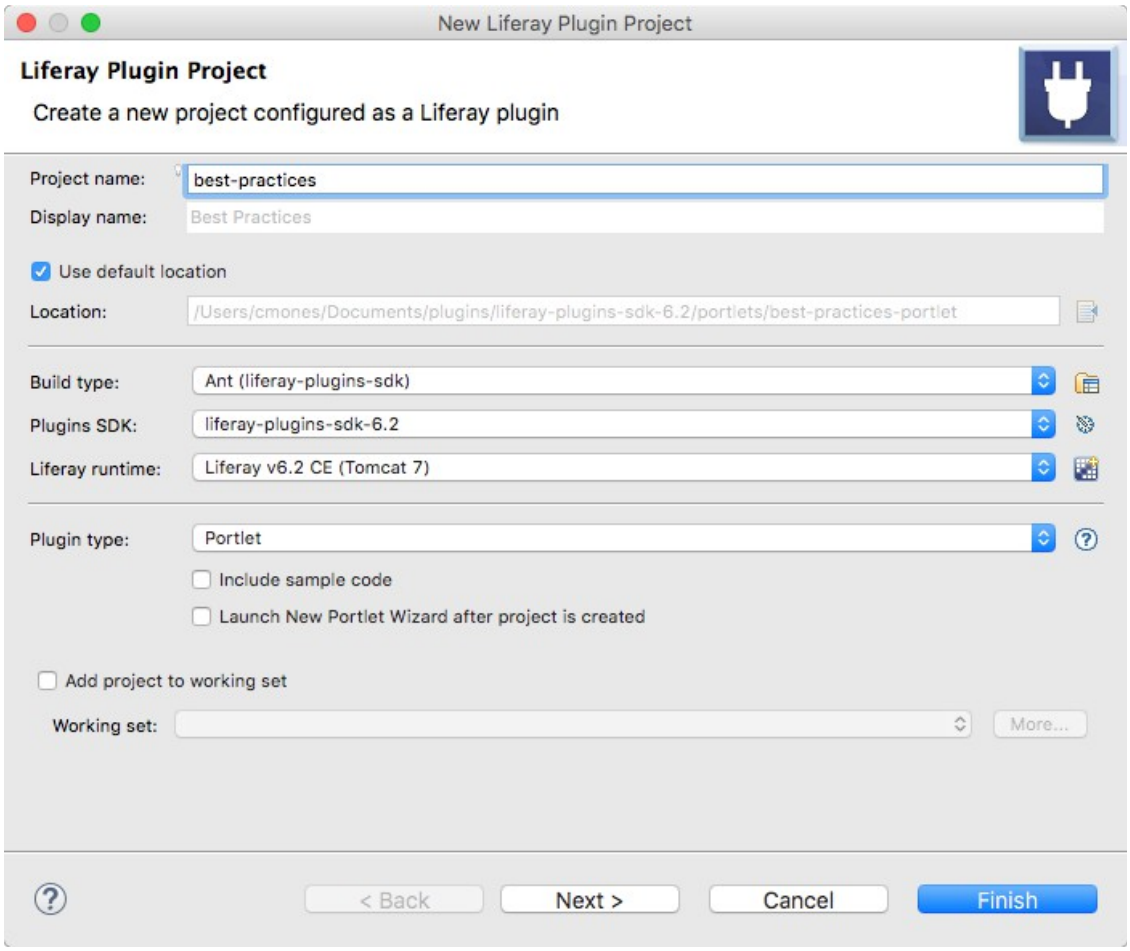


Figura 7. Assistent per a la creació de *portlets* en Liferay.

L'execució d'aquest assistent, que es dut a terme per Apache Ant, crea una nova carpeta dins del directori d'instal·lació de Liferay Plugins SDK anomenada *best-practice-portlet*. Dins d'aquesta carpeta es crea una estructura de subcarpetes que haurà de contenir tot el codi del *portlet*. Les subcarpetes i fitxers més importants són:

- *docroot*: el directori arrel de l'aplicació. Té diferents subdirectoris.
- *WEB-INF*: la carpeta estàndard *WEB-INF* per a mòduls web ja que els *portlets* són també mòduls web. Aquesta carpeta conté diversos fitxers de configuració.
- *WEB-INF/src*: el codi font del *portlet*.
- *WEB-INF/portlet.xml*: el fitxer de configuració del *portlet*. Segueix l'estàndard JSR-286 de configuració de *portlets*.
- *build.xml*: el script de construcció que Ant utilitza per a compilar i desplegar el projecte.

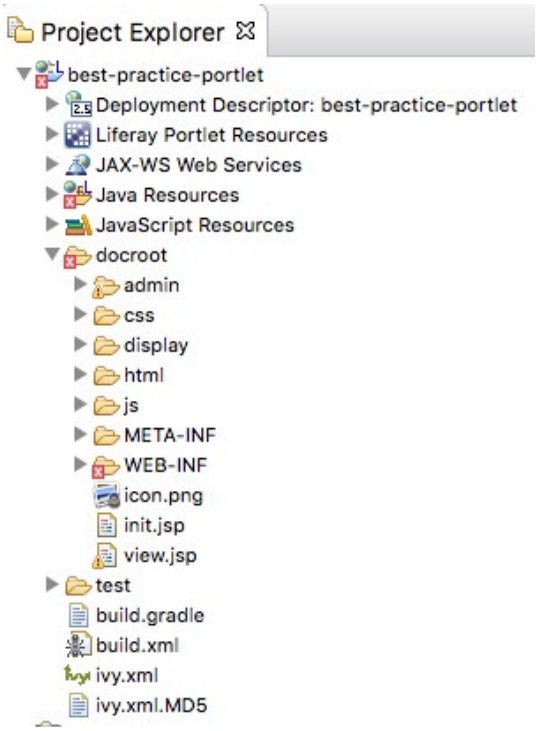


Figura 8. Estructura del projecte de *portlet*.

Obrim el fitxer *portlet.xml* que ha estat generat amb la creació del nou projecte. Veiem que la classe del *portlet* es defineix com *com.liferay.util.bridges.mvc.MVCPortlet*. Esta classe és una subclasse de *GenericPortlet* que, al seu torn, implementa la classe interfície *Javax.portlet.Portlet*. Tots els *portlets* de Liferay es basen en *MVCPortlet* perquè aquesta implementació facilita la gestió de pàgines: per a determinar quin JSP s'ha de visualitzar únicament hem de utilitzar un paràmetre de tipus render anomenat *jspPage* que establirà

la localització del JSP a mostrar.

En la fase de disseny havíem dit que caldria desenvolupar dos *portlets*: un per a l'administració de les dades de bones pràctiques i l'altre de visualització d'aquestes dades en el nostre portal web. Doncs bé, a continuació anem a configurar el nostre projecte perquè el *portlet* d'administració s'integre en el Control Panel de Liferay enlloc de fer ús del mode Edit estàndard (a través d'una icona de configuració en la finestra del *portlet*) .

El fitxer *portlet.xml* queda de la següent forma:

```
<?xml version="1.0"?>

<portlet-app xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd
http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd" version="2.0">
	<portlet>
		<portlet-name>best-practice</portlet-name>
		<display-name>Best Practice</display-name>
		<portlet-
class>org.vinaros.pathtoajob.bestpractice.portlet.BestPracticePortlet</portlet-class>
		<init-param>
			<name>view-template</name>
			<value>/view.jsp</value>
		</init-param>
		<expiration-cache>0</expiration-cache>
		<supports>
			<mime-type>text/html</mime-type>
		</supports>
		<resource-bundle>content.Language</resource-bundle>
		<portlet-info>
			<title>Best Practice</title>
			<short-title>Best Practice</short-title>
			<keywords>Best Practice</keywords>
		</portlet-info>
		<security-role-ref>
			<role-name>administrator</role-name>
		</security-role-ref>
		<security-role-ref>
			<role-name>guest</role-name>
		</security-role-ref>
		<security-role-ref>
			<role-name>power-user</role-name>
		</security-role-ref>
		<security-role-ref>
			<role-name>user</role-name>
		</security-role-ref>
	</portlet>
	<portlet>
		<portlet-name>best-practice-admin</portlet-name>
		<display-name>Best Practice Administration</display-name>
		<portlet-
class>org.vinaros.pathtoajob.bestpractice.portlet.BestPracticeAdminPortlet</portlet-class>
		<init-param>
			<name>view-template</name>
			<value>/admin/view.jsp</value>
		</init-param>
		<init-param>
			<name>add-process-action-success-action</name>
			<value>false</value>
		</init-param>
		<expiration-cache>0</expiration-cache>
		<supports>
			<mime-type>text/html</mime-type>
		</supports>
		<resource-bundle>content.Language</resource-bundle>
		<portlet-info>
			<title>Best Practice Administration</title>
```

```
                <short-title>Best Practice Administration</short-title>
                <keywords>Best Practice Administration</keywords>
            </portlet-info>
            <security-role-ref>
                <role-name>administrator</role-name>
            </security-role-ref>
            <security-role-ref>
                <role-name>guest</role-name>
            </security-role-ref>
            <security-role-ref>
                <role-name>power-user</role-name>
            </security-role-ref>
            <security-role-ref>
                <role-name>user</role-name>
            </security-role-ref>
    </portlet>
</portlet-app>
```

Per tant, es defineixen dos *portlets*, *best-practice* i *best-practice-admin*, per a la visualització i l'administració respectivament. Amb el paràmetre *view-jsp* indiquem on es troba el JSP que implementa el mode Vista del *portlets*. En el *portlet* de visualització el valor del paràmetre és */view.jsp*, és a dir, la Vista s'implementa a través del JSP *view.jsp* que està en el directori *docroot*. En el portlet d'administració el JSP de la Vista serà /admin/view.jsp, és a dir, s'implementarà a través de l'arxiu *view.jsp* que estarà en la subcarpeta *admin* dins de *docroot*.

També afegim l'atribut *resource-bundle* amb el valor *content.Language* per a indicar que els *portlets* seran multilingües.

A continuació editem el fitxer *liferay-portlet.xml* que, com *portlet.xml*, també es troba en el directori /docroot/*WEB-INF* i que s'utilitza per a descriure de quina forma els *portlets* es despleguen en Liferay. Afegim el codi següent per a indicar-li a Liferay, entre altres coses, que el *portlet* d'administració s'ha d'afegir a l'àrea Content del Control Panel:

```
    <portlet>
        <portlet-name>best-practice-admin</portlet-name>
        <icon>/icon.png</icon>
        <control-panel-entry-category>content</control-panel-entry-category>
        <control-panel-entry-weight>1.5</control-panel-entry-weight>
        <header-portlet-css>/css/best-practice-admin.css</header-portlet-css>
        <header-portlet-javascript>js/test.js</header-portlet-javascript>
    </portlet>
```

Finalment editarem el fitxer *liferay-display.xml*, també en el directori */docroot/WEB-INF*. Aquest arxiu s'utilitza per a configurar la forma en que els *portlets* apareixen en el menú Add > More de Liferay. Amb la següent configuració, indiquem que el *portlet best-practice* apareixerà en una categoria anomenada «Ajuntament de Vinaròs» i que el *portlet best-practice-admin* romandrà ocult. Així doncs, el *portlet* d'administració no es podrà afegir a

les pàgines del nostre portal i només hi podrem accedir a través del panell de control.

```
<?xml version="1.0"?>
<!DOCTYPE display PUBLIC "-//Liferay//DTD Display 6.2.0//EN" "http://www.liferay.com/dtd/liferay-display_6_2_0.dtd">

<display>
	<category name="Ajuntament de Vinaròs">
		<portlet id="best-practice" />
	</category>
	<category name="category.hidden">
		<portlet id="best-practice-admin" />
	</category>
</display>
```

5.2.2 Generació de la capa d'accés a base de dades

Com hem vist en la fase de disseny, en els nostres *portlets* intervenen dues entitats: *Organitzacio* i *BonaPractica*. La següent cosa que s'ha de fer és crear les taules de base de dades i la capa d'accés a aquestes taules per a poder emmagatzemar la informació sobre les organitzacions i les bones pràctiques del nostre portal web.

Liferay disposa d'una eina anomenada *Service Builder* que facilita al desenvolupador la generació de la capa de codi que realitza les transaccions a la base de dades.
Service Builder utilitza Hibernate com a servei de persistència d'objectes Java en base de dades i Spring com contenidor d'injecció de dependències. Veiem amb una mica més de detall aquestes dues tecnologies:

- Hibernate porta a terme el que s'anomena mapeig d'objectes relacionals (*ORM*, per les sigles en anglès) a través del qual les taules de base de dades son mapejades a objectes Java. Aquest servei permet als desenvolupadors treballar únicament amb objectes Java, amb els que estan més familiaritzats, mentre que Hibernate se n'ocupa de la persistència d'aquests objectes en la base de dades relacional.
- Spring s'encarrega de realitzar la injecció de dependències (*DI*, per les sigles en anglès) en la interacció de la nostra aplicació amb la base de dades. En moltes ocasions, la creació de nous objectes Java mitjançant un constructor depèn de la existència d'instàncies d'altres objectes. Spring allibera al programador d'haver de crear manualment tot un seguit d'objectes necessaris per a l'accés a base de dades. Spring instancia objectes i els injecta en altres objectes automàticament seguint unes regles de dependència entre objectes prèviament definides.

Service Builder utilitza un fitxer XML amb la definició de les taules de la base de dades per a generar tota la configuració d'Hibernate, tota la configuració de Spring, els mètodes de consulta de base de dades, la capa del model, els SQL necessaris per a la creació de la base de dades i tota la capa *Data Access Object (DAO)*.

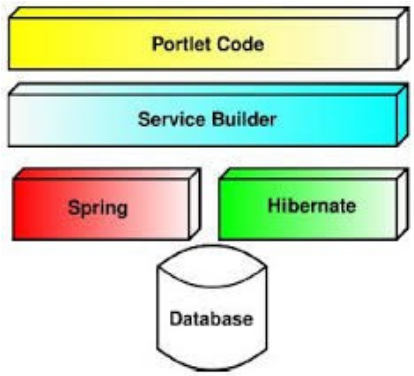


Figura 9. Esquema de *Service Builder*.

Per tant, creo un nou fitxer, anomenat *service.xml,* en la carpeta *WEB-INF* amb la informació que *Service Builder* necessita per a crear les entitats PJOrganization i PJBestPractice:

```
<?xml version="1.0"?>
<!DOCTYPE service-builder PUBLIC "-//Liferay//DTD Service Builder 6.2.0//EN"
"http://www.liferay.com/dtd/liferay-service-builder_6_2_0.dtd">

<service-builder package-path="org.vinaros.pathtoajob.bestpractice">

	<author>Cesar Mones</author>

	<namespace>PJ</namespace>

	<entity name="PJOrganization" local-service="true" remote-service="false">

		<column name="organizationId" type="long" primary="true" />

		<column name="name" type="String" />
		<column name="description" type="String" />
		<column name="contactData" type="String" />

		<column name="companyId" type="long" />
		<column name="groupId" type="long" />

		<column name="organizationBestPractices" type="Collection"
		entity="PJBestPractice" mapping-table="Organization_BestPractices"></column>

		<order by="asc">
			<order-column name="name" />
		</order>

		<finder name="G_ON" return-type="Collection">
			<finder-column name="groupId" />
			<finder-column name="name" />
		</finder>

		<finder name="GroupId" return-type="Collection">
```

```
                <finder-column name="groupId" />
            </finder>

            <finder name="CompanyId" return-type="Collection">
                <finder-column name="companyId" />
            </finder>

        </entity>

        <entity name="PJBestPractice" local-service="true" remote-service="false">

            <column name="bestPracticeId" type="long" primary="true" />

            <column name="organizationId" type="long" />

            <column name="title" type="String" />
            <column name="description" type="String" />
            <column name="areaIntervention" type="String" />
            <column name="status" type="int" />
            <column name="statusDate" type="Date" />
            <column name="statusDescription" type="String" />
            <column name="typeIntervention" type="String" />
            <column name="otherTypeIntervention" type="String" />
            <column name="interventionDescription" type="String" />
            <column name="targetGroup" type="String" />
            <column name="methodology" type="String" />
            <column name="innovativeComponents" type="String" />
            <column name="resources" type="String" />
            <column name="budget" type="String" />
            <column name="financing" type="String" />
            <column name="partners" type="String" />
            <column name="outputs" type="String" />
            <column name="evalution" type="String" />
            <column name="sustainability" type="String" />
            <column name="impact" type="String" />
            <column name="dissemination" type="String" />

            <column name="companyId" type="long" />
            <column name="groupId" type="long" />

            <order by="asc">
                <order-column name="title" />
            </order>

            <finder name="G_O" return-type="Collection">
                <finder-column name="groupId" />
                <finder-column name="organizationId" />
            </finder>

            <finder name="GroupId" return-type="Collection">
                <finder-column name="groupId" />
            </finder>

            <finder name="CompanyId" return-type="Collection">
                <finder-column name="companyId" />
            </finder>

        </entity>

</service-builder>
```

En aquest fitxer hi ha quatre seccions:

- Informació general: el paquet de Java on es generarà el codi d'accés a base de dades, *org.vinaros.pathtoajob.bestpractice*, i informació sobre l'autor que apareixerà en la documentació generada automàticament a partir del codi, Javadoc.

- Definició de les entitats: l'etiqueta *entity* serveix per a definir les entitats *PJOrganization* i *PJBestPractice*. Les definim com a entitats locals. Si haguessen de publicar-les com a *web services* s'haurien de definir com a servei remot.
- Definició de columnes: utilitzem l'etiqueta *column* per a definir les columnes de les taules que havíem establert en la fase de disseny, assignant a cada columna el tipus de dades que li correspon. A continuació comentarem algunes particularitats respecte a les columnes definides:
  - La columna que actua com a clau primària en les dues entitats s'identifica per mitjà de l'atribut *primary* amb el valor *true*.
  - Les dues entitats incorporen dues columnes, *groupId* i *companyId*, que no havien estat arreplegades en el disseny de classes. La raó d'afegir aquestes columnes es deu a que es vol fer que el *portlet* siga de tipus no instanciable, és a dir, que només hi haja una instància del *portlet* i que, per tant, només hi haja una base de dades de bones pràctiques. Però, al mateix temps, volem que cada portal i cada site o organització que es definisca tinga dades de bones pràctiques diferents. És per això que els camps *groupId* i *companyId* s'afegiran com a paràmetres internament en totes les consultes a la base de dades: així podrem diferenciar les dades de cada portal, *site* i organització.
  - En l'entitat *PJOrganization* afegim la columna *organizationBestPractices* de tipus *Collection* i en *PJBestPractice* la columna *organizationId*. El que s'aconsegueix amb aquestes columnes es establir una relació un a molts (1 a N) entre *PJOrganization* i *PJBestPractice* a través de la clau primària de l'organització i la clau externa de la bona pràctica. *Service Builder* crearà la configuració Spring que injectarà els objectes *PJBestPractice* en el mètodes de *PJOrganization* quan siga necessari. *Service Builder* també genera el mètode *getBestPractices()* en l'objecte *PJOrganization* amb el qual podrem obtenir totes les bones pràctiques associades a una determinada organització.
  - Finalment, les etiquetes *finder* faran que *Service Builder* genere automàticament els mètodes de consulta d'objectes en la base de dades utilitzant els paràmetres definits amb *finder-column*. Per exemple, el primer *finder* de *PJOrganization* recupera organitzacions per *groupId* i nom de l'organització.

Un cop ja tenim el fitxer *service.xml* executem la tasca d'Ant *build-service*. Aquesta tasca

genera fitxers font de Java en els directoris següents:

- *docroot/WEB-INF/service/org/vinaros/pathtoajob/bestpractice*: representa la capa d'interfície. Conté classes d'utilitat i interfícies. Mai s'ha de modificar manualment ja que sempre la genera íntegrament *Service Builder*.
- *docroot/WEB-INF/src/org/vinaros/pathtoajob/bestpractice*: representa la capa d'implementació de les interfícies del directori anterior.
- *docroot/WEB-INF/src/META-INF*: tota la configuració d'Hibernate i Spring.

Per tant, *Service Builder* se n'ocupa de configurar automàticament tota la capa de persistència i ens proporciona classes per a executar les funcions de base de dades que necessita la nostra aplicació.
La següent figura mostra les diferents capes que intervenen en l'accés a la base de dades. Com podem veure, es fa ús del patrons de disseny DAO i DTO per a aconseguir un disseny consistent en la capa de persistència.

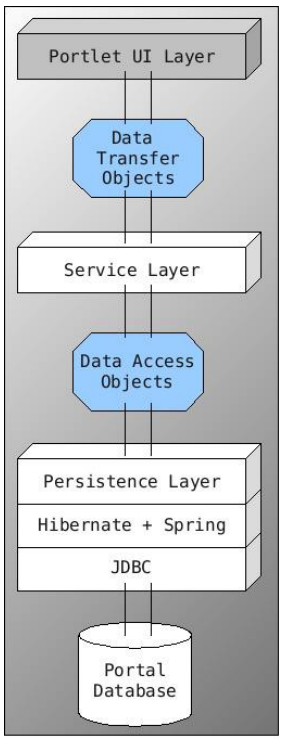


Figura 10. Esquema de capes de persistència.

Service Builder genera totes les capes automàticament per sota de la capa d'interfície d'usuari, *Portlet UI Layer*. L'únic que haurem de modificar serà la capa *DTO (Data Transfer Objects)* que és la capa que es comunica amb la capa de servei, actuant com un

buffer entre la lògica de negoci de la nostra aplicació i el codi de base de dades. Aquesta capa es correspon amb els fitxers *PJOrganizationLocalServiceImpl.java* i *PJBestPracticeLocaServiceImpl.java*, que es troben en el directori *docroot/WEB-INF/src/org/vinaros/pathtoajob/bestpractice/service/impl*. *Service Builder* crea aquests fitxers com un esquelet, amb una sèrie de mètodes que haurem d'implementar en base a la funcionalitat que necessitem. *PJOrganizationLocalServiceImpl* extén la classe *PJOrganizationLocalServiceBaseImpl* que al seu torn implementa la interfície *PJOrganizationLocalService*. Tant la superclasse com la interfície han estat íntegrament creades de forma automàtica per *Service Builder*.

Implementem el mètode *addOrganization* que agregarà una nova organització a la base de dades:

- Es crea un objecte *PJOrganization* utilitzant l'objecte *pjOrganizationPersistence* que serà injectat en aquest classe per Spring. S'usa el mètode *create*, passant-li com a paràmetre una clau primària generada amb la utilitat *Counter* de Liferay.
- Es reserven recursos per als objectes que han de ser persistits. Els recursos són necessaris per a implementar el sistema de permisos de Liferay.
- S'assignen als atributs de l'objecte *PJOrganization* que s'acaba de crear els valors que ens passen com a paràmetre.
- Es persisteix l'organització en base de dades mitjançant el mètode *update* de *pjOrganizationPersistence*.
- Es retorna el nou objecte creat a la capa que ha cridar a aquest mètode.

```
	public PJOrganization addOrganization(PJOrganization newOrganization, long userId) throws
SystemException, PortalException {

		PJOrganization organization =
pjOrganizationPersistence.create(counterLocalService.increment(PJOrganization.class.getName()));

		resourceLocalService.addResources(newOrganization.getCompanyId(),
newOrganization.getGroupId(), userId,
PJOrganization.class.getName(),organization.getPrimaryKey(),false,true,true);

		organization.setName(newOrganization.getName());
		organization.setDescription(newOrganization.getDescription());
		organization.setContactData(newOrganization.getContactData());
		organization.setCompanyId(newOrganization.getCompanyId());
		organization.setGroupId(newOrganization.getGroupId());

		return pjOrganizationPersistence.update(organization);
	}
```

De forma anàloga s'implementen mètodes per a esborrar i consultar organitzacions de la base de dades:

```
	public void deleteOrganization(long organizationId) throws NoSuchOrganizationException,
SystemException, PortalException {

		PJOrganization organization =
pjOrganizationPersistence.findByPrimaryKey(organizationId);
		deleteOrganization(organization);

	}

	public void deleteOrganization(PJOrganization organization) throws PortalException,
SystemException {

		resourceLocalService.deleteResource(organization.getCompanyId(),
PJOrganization.class.getName(), ResourceConstants.SCOPE_INDIVIDUAL, organization.getPrimaryKey());
		pjOrganizationPersistence.remove(organization);

	}


	public List<PJOrganization> getAllOrganizations (long groupId) throws SystemException {

		List<PJOrganization> organizations =
pjOrganizationPersistence.findByGroupId(groupId);
		return organizations;

	}
```

Un cop implementat la classe *PJOrganizationLocalServiceImpl* fem el mateix amb la classe *PJBestPracticeLocalServiceImpl*.

5.2.3 Creació de la interfície d'usuari del *portlet* d'administració

5.2.3.1 Estructura del *portlet*

El *portlet* d'administració haurà de permetre a l'usuari la creació, eliminació, modificació i visualització d'organitzacions i bones pràctiques. Per tant, la interfície d'usuari del *portlet* constarà dels següents elements:

- Un botó per a la creació d'organitzacions. Al prémer aquest botó es presentarà una pàgina amb un formulari per a emplenar les dades de l'organització.
- Un botó per a la creació de bones pràctiques. Al prémer aquest botó es presentarà una pàgina amb un formulari per a emplenar les dades de la bona pràctica.
- Una taula amb les organitzacions existents. En cada fila de la taula hi haurà un menú desplegable amb les opcions d'eliminació, modificació, visualització i definició de permisos de l'organització corresponent.
- Una taula amb les bones pràctiques existents. En cada fila de la taula hi haurà un menú desplegable amb les opcions d'eliminació, modificació, visualització i definició de permisos de la bona pràctica corresponent.

En el fitxer de configuració dels *portlets*, *portlet.xml*, havíem establert que la vista del *portlet* d'administració s'implementaria a través del fitxer */docroot/admin/view.jsp*. Aquest JSP contindrà la pàgina principal del *portlet* però, a més, s'hauran de crear altres JSPs per a implementar la resta de funcionalitats:

- *add_best_practice.jsp*: formulari d'alta de bones pràctiques.
- *add_organization.jsp*: formulari d'alta d'organitzacions.
- *edit_best_practice.jsp*: formulari d'edició de bones pràctiques.
- *edit_organization.jsp*: formulari d'edició d'organitzacions.
- *view_best_organization.jsp*: pàgina de visualització de bones pràctiques.
- *view_organization.jsp*: pàgina de visualització d'organitzacions.
- *admin_actions_best_practice.jsp*: fragment de codi que s'utilitza per a implementar el menú desplegable que apareix en cada fila de la taula de bones pràctiques.
- *admin_actions_organization.jsp*: fragment de codi que s'utilitza per a implementar el menú desplegable que apareix en cada fila de la taula d'organitzacions.

5.2.3.2 Implementació de la pàgina principal del *portlet*

Per tant, comencem a editar el fitxer *view.jsp*. En primer lloc creem dos botons per a l'alta d'organitzacions i bones pràctiques. En primer lloc utilitzem la utilitat *renderURL* de *MVCPortlet* que, com hem esmentat abans, facilita la gestió de pàgines del *portlet*. Aquesta utilitat ens proporciona dues URLs que apunten als JSP que gestionen l'alta d'organitzacions i bones pràctiques. Les URLs obtingudes s'assignen a la propietat *onClick* dels botons de tal forma que, al prémer-los, es fa una cridada als JSPs especificats.

```
<portlet:renderURL var="addOrganizationURL">
  <portlet:param name="jspPage" value="/admin/add_organization.jsp" />
</portlet:renderURL>

<portlet:renderURL var="addBestPracticeURL">
  <portlet:param name="jspPage" value="/admin/add_best_practice.jsp" />
</portlet:renderURL>

<aui:form
    name="fm"
    method="post">

  <aui:fieldset>

    <aui:button-row>

      <aui:button type="cancel" value="add-organization" onClick="<%= addOrganizationURL.toString()
%>" />
      <aui:button type="cancel" value="add-best-practice" onClick="<%= addBestPracticeURL.toString()
```

```
%>" />

    </aui:button-row>

  </aui:fieldset>

</aui:form>
```

En el codi es pot veure que, enlloc d'etiquetes estàndard HTML, utilitzem etiquetes *aui* per a la construcció dels formularis i dels botons. Aquestes etiquetes pertanyen a la llibreria AlloyUI, una altra eina que incorpora Liferay per a simplificar la construcció de formularis. AlloyUI és un *metaframework* d'interfícies que, mitjançant una API, ofereix múltiples components per a construir interfícies d'usuari professionals i robustes.

Per a aconseguir un codi més estructurat, creo un fitxer *init.jsp* on declararé totes les llibreries d'etiquetes i importaré totes les classes que usen els meus JSPs. D'aquesta forma només he d'afegir la línia `<%@include file="/init.jsp" %>` als JSPs per a tenir disponibles tots els recursos que es necessiten. El contingut d'aquest fitxer és:

```
<%@ taglib uri="http://java.sun.com/portlet_2_0" prefix="portlet" %>

<%@ taglib uri="http://liferay.com/tld/aui" prefix="aui" %>
<%@ taglib uri="http://liferay.com/tld/portlet" prefix="liferay-portlet" %>
<%@ taglib uri="http://liferay.com/tld/security" prefix="liferay-security" %>
<%@ taglib uri="http://liferay.com/tld/theme" prefix="liferay-theme" %>
<%@ taglib uri="http://liferay.com/tld/ui" prefix="liferay-ui" %>
<%@ taglib uri="http://liferay.com/tld/util" prefix="liferay-util" %>

<%@ page import="java.util.List" %>
<%@ page import="java.util.Calendar" %>
<%@ page import="java.util.Collections" %>
<%@ page import="com.liferay.portal.kernel.util.HtmlUtil" %>
<%@ page import="com.liferay.portal.kernel.util.ParamUtil" %>
<%@ page import="com.liferay.portal.kernel.util.CalendarFactoryUtil" %>
<%@ page import="com.liferay.portal.kernel.dao.search.ResultRow" %>
<%@ page import="com.liferay.portal.kernel.dao.search.SearchEntry" %>

<%@ page import="com.liferay.portal.kernel.exception.SystemException" %>
<%@ page import="com.liferay.portal.kernel.util.WebKeys" %>
<%@ page import="com.liferay.portal.security.permission.ActionKeys" %>
<%@ page import="com.liferay.portal.kernel.util.ListUtil" %>
<%@ page import="com.liferay.portal.service.permission.PortalPermissionUtil" %>
<%@ page import="com.liferay.portal.service.permission.PortletPermissionUtil" %>

<%@ page import="org.vinaros.pathtoajob.bestpractice.model.PJBestPractice" %>
<%@ page import="org.vinaros.pathtoajob.bestpractice.model.PJOrganization" %>
<%@ page import="org.vinaros.pathtoajob.bestpractice.service.PJBestPracticeLocalService" %>
<%@ page import="org.vinaros.pathtoajob.bestpractice.service.PJOrganizationLocalService" %>
<%@ page import="org.vinaros.pathtoajob.bestpractice.portlet.ActionUtil" %>

<%@ page import="javax.portlet.PortletURL" %>
```

A continuació anem a construir la taula d'organitzacions. Per a fer-ho utilitzarem una altra llibreria de construcció d'interfícies d'usuari específica de Liferay, *liferay-ui*, que està construïda sobre AlloyUI. Utilitzem la classe *Search Container* per a iterar sobre les organitzacions existents en base de dades. Per a cada organització es mostra el nom, la

descripció i un menú desplegable amb les opcions esmentades abans. Per a fer la consulta d'organitzacions utilitzem la classe *ActionUtil.java* que hem creat en el directori *docroot/WEB-INF/src/org/vinaros/pathtoajob/bestpractice/portlet* per a implementar funcions auxiliars del nostre *portlet*. En aquest cas la funció *getOrganizations* el que fa és obtenir el *groupId* de l'usuari que està fent ús del portal i fer la consulta en base de dades de les organitzacions associades a aquest *groupId*. Això ens mostrarà només les organitzacions del nostre *site* en cas que el portlet s'use en més d'un.

```
<liferay-ui:search-container
      emptyResultsMessage="there-are-no-organizations"
      delta="5">

    <liferay-ui:search-container-results>
    <%
    List<PJOrganization> tempResults = ActionUtil.getOrganizations(renderRequest);

    results = ListUtil.subList(
        tempResults, searchContainer.getStart(), searchContainer.getEnd());
    total = tempResults.size();

    pageContext.setAttribute("results", results);
    pageContext.setAttribute("total", total);
    %>
    </liferay-ui:search-container-results>

    <liferay-ui:search-container-row
        className="org.vinaros.pathtoajob.bestpractice.model.PJOrganization"
        keyProperty="organizationId"
        modelVar="organization">

      <liferay-ui:search-container-column-text
          name="organization-name"
          property="name" />
      <liferay-ui:search-container-column-text
          name="organization-description"
          property="description" />
      <liferay-ui:search-container-column-jsp
          path="/admin/admin_actions_organization.jsp"
          align="right" />

    </liferay-ui:search-container-row>

    <liferay-ui:search-iterator />

  </liferay-ui:search-container>
```

Per a construir el menú desplegable de cada fila de la taula es crida al JSP *admin_action_organization*. Aquest arxiu fa ús de la classe *icon-menu* de la llibreria *liferay-ui* per a construir, d'una manera molt senzilla el menú desplegable estàndard de Liferay. Primer comprova si l'usuari té permisos per a realitzar les accions de visualització, edició, eliminació i canvi de permisos sobre les organitzacions. En cas afirmatiu, inserta en el menú una nova opció que farà una crida a una *actionURL.* Les *actionURL*, a diferència de les *renderURL*, no criden directament a un JSP. El que fan és cridar a un mètode de nom igual de la classe del *portlet*. Per exemple, l'*actionURL* de nom

editOrganization crida al mètode *editOrganization* de la classe que implementa el portlet, *BestPracticeAdminPortlet* i li passa com a paràmetre la clau primària de l'organització:

```
<%@include file="/init.jsp" %>
<%
ResultRow row = (ResultRow) request.getAttribute(WebKeys.SEARCH_CONTAINER_RESULT_ROW);
PJOrganization myOrganization = (PJOrganization) row.getObject();
long groupId = themeDisplay.getLayout().getGroupId();
String name = PJOrganization.class.getName();
String primKey = String.valueOf(myOrganization.getPrimaryKey());
%>

<liferay-ui:icon-menu>

  <c:if test="<%= permissionChecker.hasPermission(groupId, name, primKey, ActionKeys.VIEW) %>">
    <portlet:actionURL name="viewOrganization" var="viewURL">
      <portlet:param name="resourcePrimKey" value="<%= primKey %>" />
    </portlet:actionURL>

    <liferay-ui:icon image="view" message="View" url="<%= viewURL.toString() %>" />
  </c:if>

  <c:if test="<%= permissionChecker.hasPermission(groupId, name, primKey, ActionKeys.UPDATE) %>">
    <portlet:actionURL name="editOrganization" var="editURL">
      <portlet:param name="resourcePrimKey" value="<%= primKey %>" />
    </portlet:actionURL>

    <liferay-ui:icon image="edit" message="Edit" url="<%= editURL.toString() %>" />
  </c:if>

  <c:if test="<%= permissionChecker.hasPermission(groupId, name, primKey, ActionKeys.DELETE) %>">
    <portlet:actionURL name="deleteOrganization" var="deleteURL">
      <portlet:param name="resourcePrimKey" value="<%= primKey %>" />
    </portlet:actionURL>

    <liferay-ui:icon-delete url="<%= deleteURL.toString() %>" />
  </c:if>

  <c:if test="<%= permissionChecker.hasPermission(groupId, name, primKey, ActionKeys.PERMISSIONS)
%>">
    <liferay-security:permissionsURL
        modelResource="<%= PJOrganization.class.getName() %>"
        modelResourceDescription="<%= myOrganization.getName() %>"
        resourcePrimKey="<%= primKey %>"
        var="permissionsURL" />

    <liferay-ui:icon image="permissions" url="<%= permissionsURL.toString() %>" />
  </c:if>
</liferay-ui:icon-menu>
```

La següent figura mostra la pàgina principal del portlet d'administració, integrada en el Control Panel de Liferay:

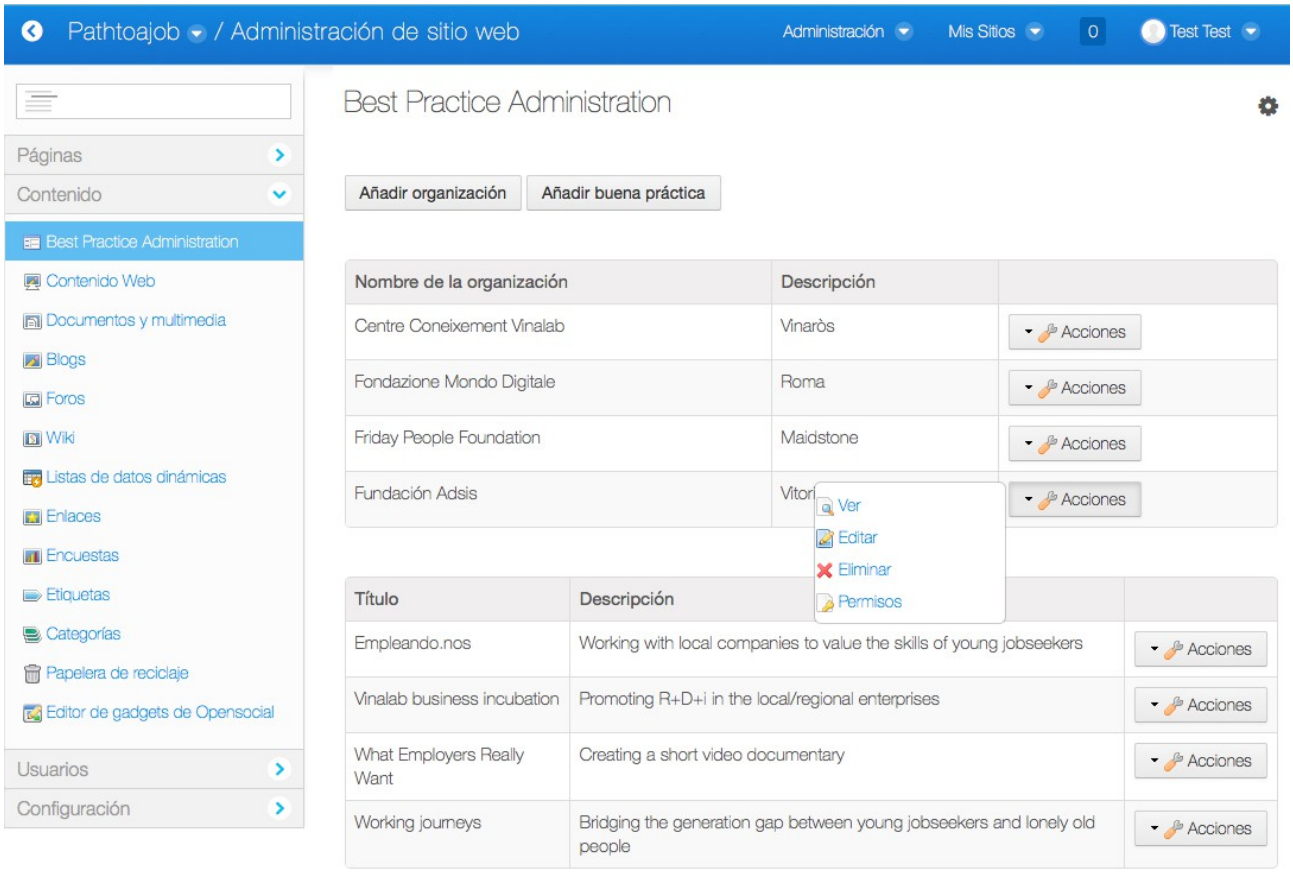


Figura 11. Pàgina principal del *portlet* d'administració.

5.2.3.3 Implementació de les accions sobre organitzacions

Si l'usuari selecciona l'opció d'edició d'organitzacions, s'executa el mètode del mateix nom del *portlet*. La classe *BestPracticeAdminPortlet.java*, que implementa el *portlet* es troba en el directori *docroot/WEB-INF/src/org/vinaros/pathtoajob/bestpractice/portlet* i conté tots s els mètodes que els JSPs executen a través dels *actionURL*. El mètode d'edició d'organitzacions comprova que la clau primària siga vàlida i, si ho és, obté un objecte *PJOrganization* utilitzant la capa DTO (*PJOrganizationLocalServiceUtil*) que hem vist a l'apartat d'accés a base de dades. Finalment, crida al JSP *edit_organization* passant-li com a paràmetre el *PJOrganization* obtingut. Aquest JSP presentarà la pàgina on l'usuari podrà introduir els canvis en l'organització.

```
public void editOrganization(ActionRequest request, ActionResponse response)
    throws Exception {

    long organizationKey = ParamUtil.getLong(request, "resourcePrimKey");

    if (Validator.isNotNull(organizationKey)) {
        PJOrganization organization =
            PJOrganizationLocalServiceUtil.getPJOrganization(organizationKey);
```

```
            request.setAttribute("organization", organization);
                    response.setRenderParameter("jspPage", editOrganizationJSP);
        }
    }
```

Veiem que fa l'arxiu *edit_organization.jsp*: *en* primer lloc empra novament les utilitats *renderURL* i *actionURL* de *MVCPortlet* per a obtenir les URLs que s'associaran als botons del formulari S*ubmit* i C*ancel*. Després construeix un formulari amb els camps de l'organització informats amb els valors del *PJOrganization* que s'ha rebut com a paràmetre. Si es prem el botó *Cancel* tornem a la pàginia inicial del *portlet*, *view.jsp.* Si, en canvi, es prem el botó *Submit* s'executa el mètode *updateOrganization* del *portlet*:

```
<%@include file="/init.jsp" %>

<%
PJOrganization organization = (PJOrganization) request.getAttribute("organization");
%>

<portlet:renderURL var="cancelURL">
  <portlet:param name="jspPage" value="/admin/view.jsp" />
</portlet:renderURL>

<portlet:actionURL name="updateOrganization" var="updateOrganizationURL" />

<h3><liferay-ui:message key="edit-organization" /></h3>

<aui:form
    name="fm"
    action="<%= updateOrganizationURL.toString() %>"
    method="post">

  <aui:fieldset>

    <aui:input
        name="resourcePrimKey"
        value="<%= organization.getOrganizationId() %>"
        type="hidden"
    />
    <aui:input
        name="organization-name"
        value="<%= organization.getName() %>"
        size="45"
    />
    <aui:input
        name="organization-description"
        value="<%= organization.getDescription() %>"
        size="45"
    />
    <aui:input
        name="organization-contact-data"
        value="<%= organization.getContactData() %>"
        size="45"
    />

    <aui:button-row>

      <aui:button type="submit"/>
      <aui:button
          type="cancel"
          value="Cancel"
          onClick="<%= cancelURL.toString() %>"
    />

    </aui:button-row>

  </aui:fieldset>
```

```
</aui:form>
```

La pantalla d'edició d'organitzacions queda així:

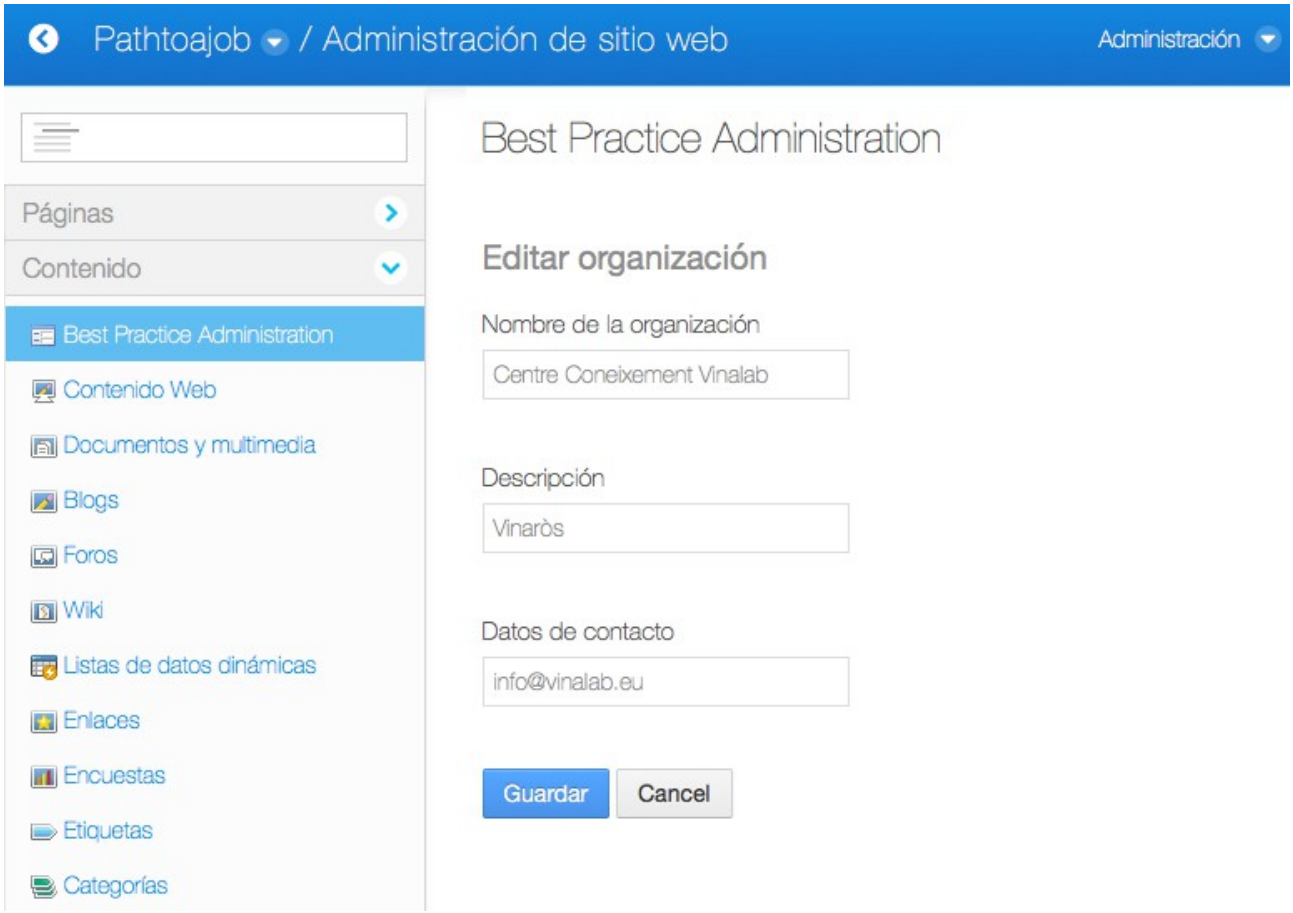


Figura 12. Pantalla d'edició d'organitzacions.

El mètode *updateOrganization* de *BestPracticeAdminPortlet* valida els camps introduïts en el formulari i, si està tot correcte, utilitza novament la capa DTO per a persistir els canvis en base de dades (a través del mètode *updatePJOrganization* d'un objecte de tipus *PJOrganizationLocalServiceUtil*). Finalment es mostra un missatge de confirmació de la modificació.

```
public void updateOrganization(ActionRequest request, ActionResponse response)
    throws Exception {

    long organizationKey = ParamUtil.getLong(request, "resourcePrimKey");
    ArrayList<String> errors = new ArrayList();
    if (Validator.isNotNull(organizationKey)) {
        PJOrganization organization =
```

```
            PJOrganizationLocalServiceUtil.getPJOrganization(organizationKey);
        PJOrganization requestOrganization = ActionUtil.organizationFromRequest(request);

        if (BestPracticeValidator.validateOrganization(requestOrganization, errors)) {
            organization.setName(requestOrganization.getName());
            organization.setDescription(requestOrganization.getDescription());
            organization.setContactData(requestOrganization.getContactData());
            PJOrganizationLocalServiceUtil.updatePJOrganization(organization);
            SessionMessages.add(request, "organizationUpdated");

        }
        else {
            for (String error : errors) {
                SessionErrors.add(request, error);

            }

        }

    }
    else {
        SessionErrors.add(request, "error-updating");
    }

}
```

Utilitzem un procediment quasi idèntic per a implementar la visualització d'organitzacions. En el cas de l'eliminació d'organitzacions la programació varia lleugerament: quan es selecciona l'opció en el menú s'executa el mètode *deleteOrganization* del *portlet* però, com que l'únic que s'ha de fer és eliminar la informació en base de dades, no es crida a un altre JSP sinò que s'executa directament el mètode *deleteOrganization* de *PJOrganizationLocalServiceUtil* per a eliminar en base de dades el registre.

5.2.3.4 Implementació de les accions sobre bones pràctiques

El procediment d'implementació de les accions sobre les bones pràctiques és anàleg al que s'ha descrit per a les organitzacions.

Les úniques particularitats que el diferencien de l'anterior es troben en els formularis d'alta, edició i visualització de bones pràctiques:

- Com que una bona pràctica sempre està associada a una organització prèviament creada els formularis d'alta i d'edició de bones pràctiques han de mostrar un selector d'organitzacions. L'usuari seleccionarà a través d'una llista desplegable l'organització que assigna a la bona pràctica. En la taula de base de dades de bones pràctiques es guardarà la clau primària de l'organització a la que pertany. La pàgina de visualització de bones pràctiques haurà de fer una consulta prèvia en base de dades a través d'aquesta clau primària per a mostrar la informació de l'organització.
- El formulari de bones pràctiques incorpora alguns camps de tipus específics com

dates i llistes desplegables.

El codi següent mostra la forma en que s'ha implementat aquests camps en el formulari d'alta de bones pràctiques. Es pot veure com es construeix la llista d'organitzacions mitjançant la classe *select* de la llibreria AlloyUI a partir de llista d'objectes de tipus *PJOrganization* que ens proporciona la capa DTO.

```
<%
List<PJOrganization> organizations =
PJOrganizationLocalServiceUtil.getAllOrganizations(themeDisplay.getScopeGroupId());
%>
<aui:select name="best-practice-organization-id">
	<aui:option value="-1">
		<liferay-ui:message key="please-choose" />
	</aui:option>
	<%
	for (PJOrganization organization : organizations) {
		%>
		<aui:option value="<%= organization.getOrganizationId() %>"><%=
organization.getName() %></aui:option>
		<%
	}
	%>
</aui:select>

<aui:input name="best-practice-status-date" />

<aui:select name="best-practice-status">
	<aui:option value="-1">
		<liferay-ui:message key="please-choose" />
	</aui:option>
	<aui:option value="0">Starting</aui:option>
	<aui:option value="1">On going</aui:option>
	<aui:option value="2">Concluded</aui:option>
</aui:select>
```

El camp *best-practice-status-date* es mostra com un selector de tipus data. Això és possible perquè, en el moment de la construcció del formulari, es consulta el fitxer *docroot/WEB-INF/src/META-INF/portlet-model-hints.xml* (també generat per *Service Builder*) que determina com es presenten les dades a l'usuari. En aquest fitxer apareixen totes les entitats que ha construït *Service Builder* amb tots els seus camps. Aquest fitxer es pot modificar per a, per exemple, canviar les dimensions d'un camp de text, la longitud màxima que podem escriure, etc. Després de modificar el fitxer s'haurà d'executar *Service Builder* i desplegar el *portlet* un altre cop.

5.2.3.5 Multilingüisme

El nostre portal ha ser multilingüe i, per tant, tots els textos que apareixen en els *portlets* s'han de mostrar en diferents idiomes segons la preferència de l'usuari del portal. Per a

aconseguir-ho hem de substituir els literals que apareixen en els *portlets* per etiquetes que seran traduïdes en temps d'execució. Liferay per la traducció uns fitxers anomenats *Language.properties* amb un codi de dues lletres dels diferents idiomes. Per exemple, si, com en el nostre cas, volem que el nostre portal estiga traduït a l'anglès, castellà i italià, haurem de crear els fitxers *Language_en.properties*, *Language_es.properties* i *Language_it.properties* i ubicar-los en el directori *docroot/WEB-INF/src/content*.

El fitxer *Language_es.properties* queda de la següent forma:

```
##
## Messages
##

Organization=Organización
action.ADD_ORGANIZATION=Añadir organización
model.resource.org.vinaros.pathtoajob.bestpractice.model.PJOrganization=Organización
model.resource.org.vinaros.patjtoajob.bestpractice.model.PJBestPractice=Buena Práctica
View=Ver
Edit=Editar
please-choose=por favor, elija

# Organization Form Messages
add-organization=Añadir organización
view-organization=Ver organización
edit-organization=Editar organización
organization-saved-successfully=Organización guardada con éxito
organizationDeleted=La organización ha sido borrado con éxito
organizationUpdated=La organización ha sido modificada con éxito
there-are-no-organizations=No hay organizaciones

# Organization Form Fields
organization-contact-data=Datos de contacto
organization-description=Descripción
organization-name=Nombre de la organización

# Best Practice Form Messages
add-best-practice=Añadir buena práctica
view-best-practice=Ver buena práctica
edit-best-practice=Editar buena práctica
display-best-practices=Buenas Prácticas
display-best-practice=Buena Práctica
there-are-no-best-practices=No hay buenas prácticas

# Best Practice Form Fields
best-practice-description=Descripción
best-practice-title=Título
best-practice-organization=Organización
best-practice-organization-id=Organización
best-practice-area-intervention=Área de intervención
best-practice-status=Estado
best-practice-status-date=Fecha de estado
best-practice-status-description=Descripción del estado
best-practice-type-intervention=Tipo de intervención
best-practice-other-type-intervention=Otro tipo de intervención
best-practice-intervention-description=Descripción de la intervención
best-practice-target-group=Grupo objetivo
best-practice-methodology=Metodología
best-practice-innovative-components=Componentes innovadores
best-practice-resources=Recursos
best-practice-budget=Presupuesto
best-practice-financing=Financiación
best-practice-partners=Socios
best-practice-outputs=Resultados
best-practice-evaluation=Evaluación
best-practice-sustainability=Sostenibilidad
best-practice-impact=Impacto
best-practice-dissemination=Diseminación
```

Si es vol posar un missatge en qualsevol *portlet* es fa de la següent forma:

```
<liferay-ui:message key="add-organization" />
```

Es consulta la clau en el fitxer associat a l'idioma seleccionat i es presenta la traducció. El mateix ocorre amb els elements AlloyUI que, o bé per l'atribut *name* o bé per l'atribut *value*, també són traduïts.

```
<aui:input name="organization-name" size="45" />
```

Els *portlets* també hereten els fitxers de traducció dels portal per la qual cosa no s'han de traduir elements estàndard, com per exemple l'etiqueta del botó *Back*.

```
<aui:button
  type="submit"
  value="back"
  onClick="<%= okURL.toString() %>"
/>
```

5.2.4 Creació de la interfície d'usuari del *portlet* de visualització

5.2.4.1 Estructura del *portlet*

El *portlet* de visualització és molt senzill. Consta d'una pàgina principal on es mostra una llista de bones pràctiques dutes a terme per cada organització. El nom de cada bona pràctica es presenta com un enllaç a un altra pàgina. Quan l'usuari del portal prem l'enllaç d'una bona pràctica de la llista es mostra una altra pàgina en la que apareix el detall de la bona pràctica. Prement un botó tornarem a la pàgina principal del *portlet*.

En el fitxer de configuració dels *portlets*, *portlet.xml*, havíem establert que la vista del *portlet* de visualització s'implementaria a través del fitxer */docroot/view.jsp*. Aquest JSP contindrà la pàgina principal del *portlet* però, a més, s'haurà de crear un altre JSP per a implementar la pàgina de visualització del detall de bones pràctiques. L'anomenarem *display_best_practice.jsp*.

5.2.4.2 Implementació de la pàgina principal del portlet

La construcció de la llista del bones pràctiques és fa de la següent forma: es realitza una consulta de totes les organitzacions i, després, s'itera sobre elles. Per cada iteració es mostra el nom de l'organització i es fa una consulta de totes les seues bones pràctiques. S'itera un segon cop sobre les bones pràctiques de l'organització mostrant el nom de la bona pràctica en forma d'enllaç.

En cada enllaç s'inclou una URL construïda amb la utilitat *renderURL* de *MVCPortlet.* Aquí no utilitzem únicament el paràmetre *jspPage* per a construir la URL sinó que també incorporem la clau primària de la bona pràctica. D'aquesta forma, la pàgina destí, *display_best_practice.jsp*, sabrà quina bona pràctica ha de mostrar.

```
<%@include file="init.jsp" %>

<h3><liferay-ui:message key="display-best-practices" /></h3>

	<table>
    <%
    List<PJOrganization> organizations = ActionUtil.getOrganizations(renderRequest);
	String primKey;

	for (PJOrganization organization : organizations) {
		%>
		<tr><td colspan="2"><h4><strong><%= organization.getName() %></strong></h4></td></tr>
		<%
		List<PJBestPractice> bestPractices =
ActionUtil.getBestPracticesByOrganization(renderRequest, organization.getOrganizationId());
		for (PJBestPractice bestPractice : bestPractices) {
			primKey = String.valueOf(bestPractice.getPrimaryKey());
				%>
				<tr><td width="30%" /><td width="70%">
					<portlet:renderURL var="displayBestPracticeURL">
						<portlet:param name="jspPage"
value="/display/display_best_practice.jsp" />
						<portlet:param name="bestPracticeId" value="<%= primKey
%>" />
					</portlet:renderURL>
					<a href="<%= displayBestPracticeURL.toString() %>"><%=
bestPractice.getTitle() %></a></td></tr>
				<%
			}
		%>
		<tr><td> </td></tr>
		<%
	}
	%>
	</table>
```

La vista principal del *portlet* queda de la següent forma:

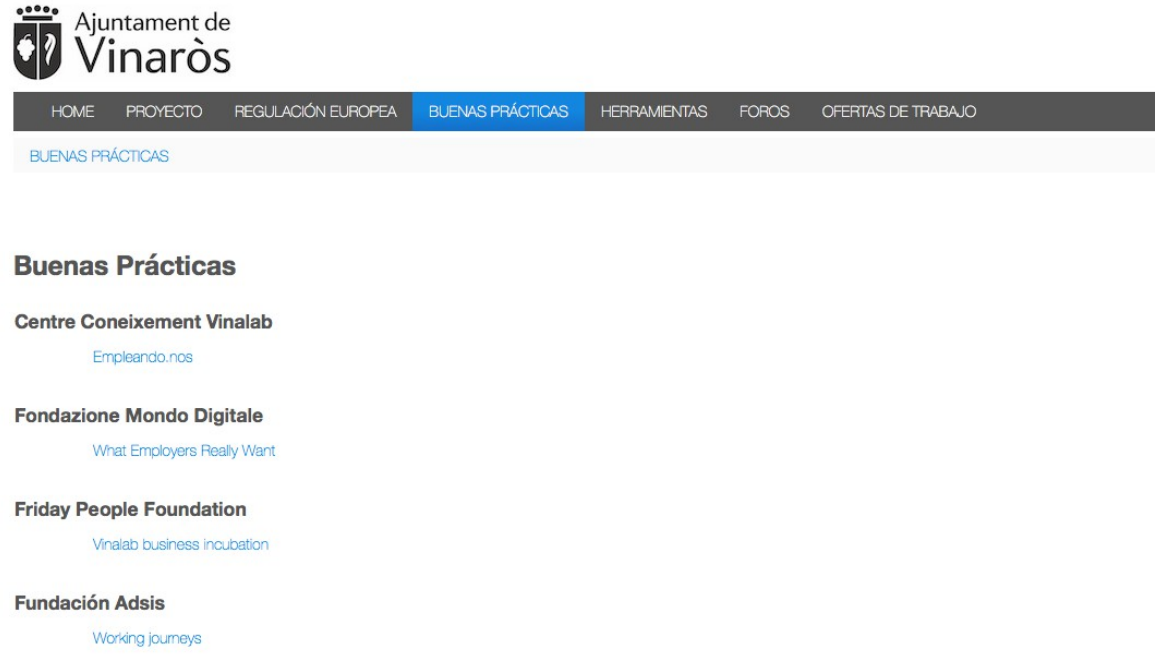


Figura 13. Vista principal del *portlet* de visualització.

El detall de la bona pràctica, aquest cop en la versió anglesa, es mostra així:

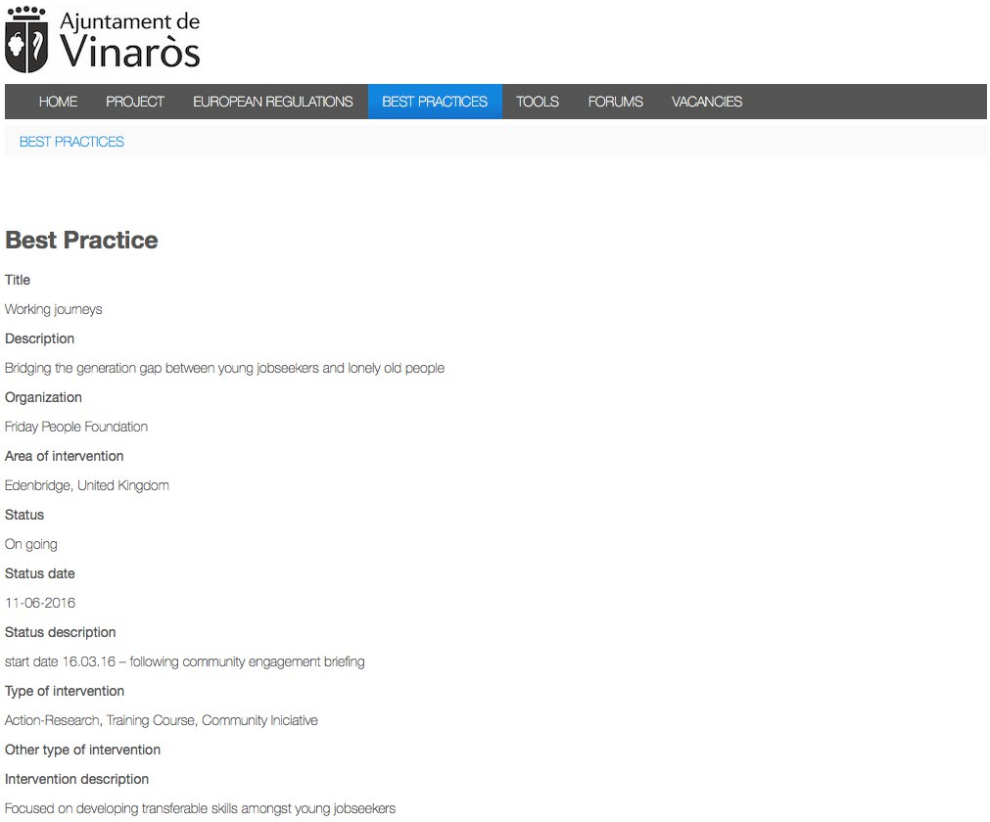


Figura 14. Pantalla del detall d'una bona pràctica.

5.3 Creació del portal web

Les imatges anteriors mostren l'aspecte general del portal que hem construït. La construcció de portals amb *plugins* estàndard de Liferay és molt senzilla ja tot el procés es fa de forma interactiva a través del panell de control.

A grans trets, els passos que hem portat a terme són els següents:

- Creació d'un nou *site* i personalització del logo.
- Configuració dels idiomes del portal.
- Creació de les pàgines del portal. S'han de crear per als tres idiomes. En la imatge següent es mostra les pàgines de la versió en castellà.
- Addició dels *portlets* a les diferents pàgines segons el que s'havia determinat en la fase de disseny. Lògicament, en la pàgina bones pràctiques afegim el *portlet* de visualització de bones pràctiques que hem desenvolupat.
- Configuració dels *portlets* estàndard amb les nostres preferències.

Figura 15. Estructura de pàgines del portal web.

A continuació es pot veure l'aspecte de la pàgina *home* del nostre portal amb els *portlets* de *Blogs*, *Calendar* i selector d'idioma.

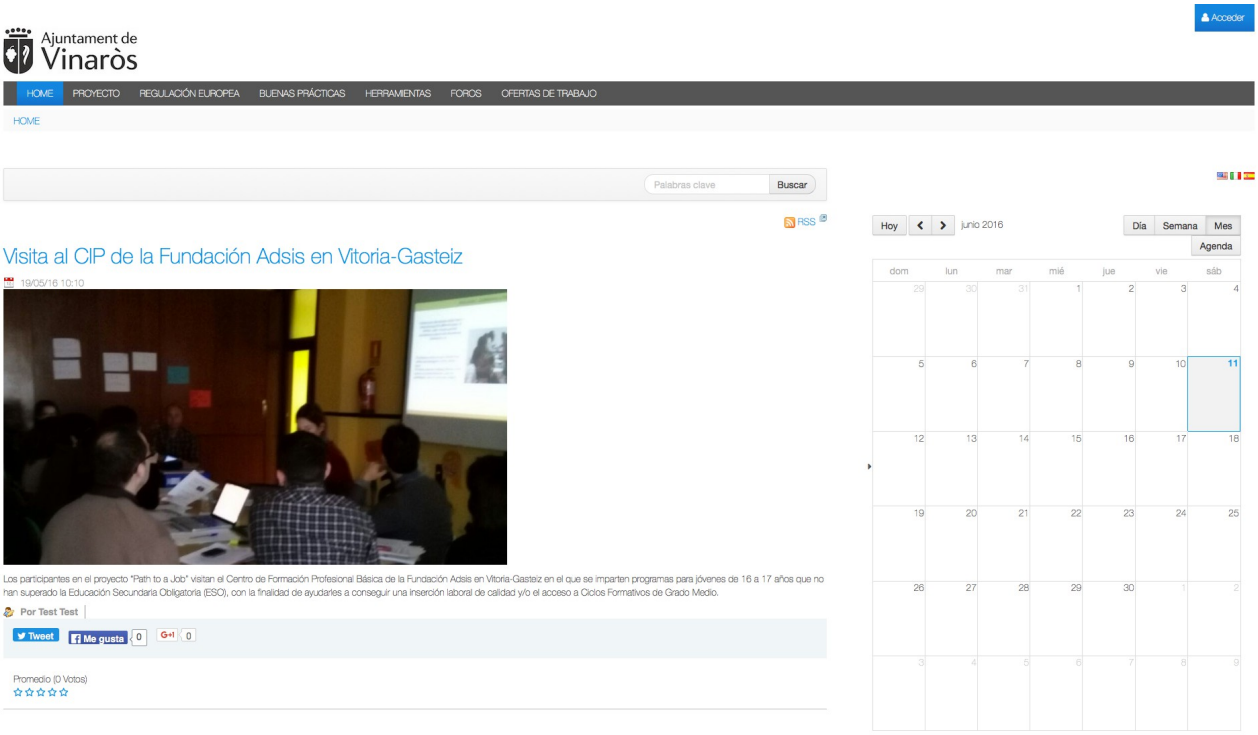


Figura 16. Pàgina *home* del portal web.

6 Conclusions

L'objectiu principal del projecte era analitzar el potencial de Liferay per a la construcció i gestió dels continguts de portals web. Es tractava de determinar si Liferay representava una bona opció en un moment en que el departament d'Informàtica de l'Ajuntament de Vinaròs s'està plantejant implantar un sistema de gestió de continguts per a unificar tots els webs municipals.
Per tant, el plantejament inicial era simple: provar Liferay mitjançant el desenvolupament d'un web real que ens havia sol·licitat el departament de Benestar Social de l'Ajuntament.

Per tal d'aconseguir l'objectiu del projecte ha sigut necessària una fase inicial d'aprenentatge de la plataforma Liferay. He adquirit sòlids coneixements de diferents aspectes d'aquest producte:

- Instal·lació d'un entorn de desenvolupament basat en Eclipse i integrat amb el servidor Liferay.
- Desenvolupament de *portlets* a través del *framework* MVC de Liferay.
- Desenvolupament de portals web i gestió de continguts per mitjà de Liferay Portal.

Així doncs, després de la feina realitzada, he arribat a la conclusió de que Liferay és una molt bona opció a tenir en compte a l'hora d'implantar un gestor de continguts ja que aporta grans avantatges en diferents àmbits del desenvolupament web: generació de la capa d'accés a base de dades, construcció d'interfícies d'usuari, simplicitat del seu *framework* MVC, facilitat en la construcció de portals a través dels més de 60 *portlets* que integra el seu nucli, etc.

7 Bibliografia

[1] Liferay Developer Tutorials: https://dev.liferay.com/develop/tutorials.

[2] Using Liferay Portal 6.2: https://dev.liferay.com/documents/10184/510059/indexed-using-liferay-portal-62.pdf.

[3] Sezov, R. (2012). Liferay in action. Manning Publications Co.

[4] Duckett, J. (2011). HTML & CSS. John Wiley & Sons, Inc.

[5] Duckett, J. (2014). Javascript & Jquery. John Wiley & Sons, Inc.

[6] Flanagan, D (1997). Java in a Nutshell. O'Reilly Media.

[7] Sarin, A. (2012). Portlets in action. Manning Publications Co.