Portlets de publicación y registro de eventos: Event Registration

Ana Mª Mendoza Guasch
Grado en Ingeniería Informática
Área TFG Java EE

Albert Grau Perisé
Santi Caballe Llobet

14/06/2016

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Portlets de publicación y registro de eventos: Event Registration*
Nombre de la autora:	*Ana Mª Mendoza Guasch*
Nombre del consultor:	*Albert Grau Perisé*
Nombr del PRA:	*Santi Caballe Llobet*
Fecha de entrega:	*06/2016*
Titulación o programa:	*Grado en Ingeniería Informática*
Área del Trabajo Final:	*Java EE*
Idioma del trabajo:	*Castellano*
Palabras clave:	*Liferay, portlet, Java EE*

Resumen del Trabajo:

En este proyecto se desarrolla una aplicación de gestión y registro en eventos en Java EE. Para su implementación se utiliza la tecnología de portlets en su especificación JSR286 implementada para Liferay Portal. Se han utilizado las librerías propias de Liferay en todas las capas de la aplicación. La generación del esquema en bases de datos y las clases de persistencia se han creado con Service Builder, una tecnología de generación de código que incorpora el SDK de Liferay y que utiliza Spring e Hibernate. En la capa de presentación se utiliza JSP, los taglibs propios de Liferay, AlloyUI y Bootstrap.

Se trata de un proyecto de código abierto, disponible públicamente en GitHub y extensible.

El proyecto resultante consta de dos portlets: un portlet para la inscripción pública y otro para la administración de los eventos y los registros de usuario. Aunque la administración se podría haber separado en distintos portlets, se ha creído más conveniente mantener toda la administración dentro de una misma opción del panel de control, de la misma forma que ocurre con otros portlets ya disponibles en Liferay.

La implementación no ha resultado especialmente compleja aunque sí que ha presentado una dificultad mayor debido al uso extensivo de formatos de tipo fecha. Ha requerido una gran dedicación de tiempo y ha sido posible gracias a todo lo aprendido anteriormente en el Grado y a que ya disponía de experiencia previa en el desarrollo de portlets para Liferay.

Abstract:

In this project I develop an event management and registration application in Java

EE. It was implemented using the portlet technology in its JSR286 specification implemented for Liferay Portal. I have used Liferay's own libraries in all application layers. The model schema and persistence classes have been created with Service Builder, a code generation tool that provides Liferay Portal SDK that makes use of Spring and Hibernate. In the presentation layer, I have used JSP, Liferay's own taglibs, AlloyUI and Bootstrap.

This is an open source and extensible project, publicly available on GitHub.

The resulting project consists of two portlets: one portlet offers the public registration and the other provides event and registration administration. Although the administration portlet could have been split into different portlets, I though it was more convenient to keep all administration functionalities in the same control panel option, just like other portlets already available in Liferay.

The implementation has not been particularly complex, although certain difficulty was introduced due to the use of the extensive use of data type formats. It required a great commitment of time and it has been possible thanks to all I learned on this degree and my previous experience developing portlets for Liferay Portal.

Índice

Objetivo del proyecto

El objetivo de este proyecto final de grado ha sido desarrollar una aplicación basada en JavaEE poniendo en práctica todas las fases del desarrollo de un producto software. Para su implementación he escogido Liferay Portal y la tecnología de portlets.

Se trata de una aplicación que ya se podría desplegar y utilizar en un entorno real, pero a la que también se le pueden incluir muchas nuevas funcionalidades y mejoras. Por ello, se ha desarrollado una aplicación de código libre, disponible en un repositorio público de GitHub: https://github.com/melfina/EventReg

El proyecto desarrollado en este trabajo de final de grado se basa en un supuesto ficticio que se presenta en los siguientes apartados. También se incluye toda la documentación generada en las distintas fases del proyecto: toma de requisitos, análisis funcional, diseño lógico, prototipos y planificación.

Descripción

Se ha desarrollado un sistema de registro en eventos que permite gestionar distintos eventos y que los visitantes del portal se registren en ellos.

Se deseaba que el componente resultante sea lo más flexible y modular posible, de manera que pueda ajustarse a distintos portales y se pueda distribuir a través del *Liferay Marketplace*. Esto implica que se ha de implementar necesariamente en ingles, por lo que se la aplicación estará disponible en tres idiomas: español, catalán e inglés.

Además de implementar las funcionalidades deseadas, la aplicación también debe tener una interfaz atractiva e intuitiva. También se han tenido en cuenta otras cuestiones inherentes al desarrollo de portales web como el SEO y la accesibilidad.

Finalmente, se ha procurado, durante la implementación del proyecto, hacer uso de buenas prácticas en el desarrollo de aplicaciones, mediante el uso de un sistema de control de versiones y escribiendo código legible intentando facilitar su mantenimiento.

Motivación

La idea del proyecto parte de la necesidad detectada en mi trabajo actual como desarrolladora de portales para la administración pública. Aunque Liferay Portal ya dispone de un portlet de gestión de eventos, sus funcionalidades están enfocadas para su uso como agenda personal o de equipos. En ocasiones, resulta necesario publicar eventos en los que pueden participar ciudadanos o colectivos y que éstos puedan inscribirse. Un caso más concreto de esta necesidad, es la publicación de jornadas o seminarios. Cuando el aforo es limitado, no es sólo necesario publicar su información, si no también facilitar a los ciudadanos su inscripción en el mismo y a los gestores, la gestión de éstas inscripciones.

Toma de requisitos

Para la creación de nuestro proyecto, un cliente nos contacta con el siguiente correo:

“Necesitamos incorporar en nuestro portal la posibilidad de publicar eventos y que nuestros visitantes puedan registrarse en ellos. El sistema ha de ser lo suficientemente flexible como para que nos permita publicar distintos tipos de eventos, como pueden ser conferencias o sesiones de networking que pueden ser gratuitas o de pago. Además, hemos de poder establecer un número máximo de plazas y gestionar los registros. Para algunos eventos gratuitos el registro no puede ser automático, si no que necesitamos poder revisarlos antes de que se apruebe. También hemos de poder añadir distintos tipos de opciones de registro para algunos eventos, como puede ser el ‘registro de evento’ o ‘registro sólo un día’ para las personas que sólo acudirán a un evento en uno de los días que se celebra.

Todavía no sabemos muy bien si queremos que los usuarios se tengan que registrar en el portal o no, ya que queremos que el registro sea lo más sencillo posible. Lo que sí necesitamos es poder mantener un listado de usuarios que puedan dar de alta, editar y gestionar los eventos y sus asistentes. Necesitaremos poder ver estadísticas e informes globales sobre el progreso de registro en los eventos y también individuales (para cada evento). Además, queremos poder exportar los listados de usuarios registrados en un evento, para poder gestionarlo en el momento del evento.

En cuanto al funcionamiento de cara a los visitantes, cada uno de los eventos listados en el portal ha de permitir que se comparta en redes sociales. Queremos también poder incluir fotos e información básica del evento como su nombre y descripción, dónde se llevará a cabo, fecha y hora, número de plazas totales y disponibles. Para algunos eventos, necesitaremos limitar la fecha límite de registro para que no puedan registrarse hasta el último momento.

Sería perfecto que pudiéramos incluir un mapa interactivo marcando el lugar donde se va a llevar a cabo el evento y que los usuarios pudieran imprimir la información del evento y de su registro.

Por último, necesitaremos poder acceder a los datos de eventos y registros ya que queremos poder implementar el sistema con el ERP de nuestra empresa.”

Especificación de las funcionalidades

Según los requisitos que nos indicaron por correo y diversas reuniones que se llevaron a cabo posteriormente, identificamos los siguientes subsistemas:

- Gestión de usuarios
- Gestión de eventos
- Gestión de reservas

Subsistema de gestión de usuarios

El subsistema de gestión de usuarios se encargará de la identificación, registro y gestión de permisos de los usuarios encargados de la gestión de eventos.

Alta de usuario

Crea un nuevo usuario que podrá identificarse en el sistema. Se creará como miembro de uno o varios sites y se le asignarán una serie de roles que a su vez tienen asignados varios permisos.

Asignar permisos

Permite asignar permisos a un rol de forma que los usuarios que tengan asignado un rol tengan acceso a distintas funcionalidades.

Identificación de usuario

Permite a un usuario dado de alta identificarse en el portal con los roles y accesos que tenga asignados.

Editar usuario

Muestra un formulario con los datos actualmente guardados para el usuario y permite modificarlos.

Subsistema de gestión de eventos

El subsistema de gestión de eventos proporcionará todas las funcionalidades de mantenimiento y visualización de eventos.

Alta de evento

Permite dar de alta un nuevo evento por parte de un gestor con permisos de alta de evento.

Modificación de evento

Permite modificar un evento ya dado de alta por parte de un gestor con permisos de modificación sobre los eventos.

Baja de evento

Permite eliminar un evento ya dado de alta por parte de un gestor con permisos de eliminación sobre los eventos.

Búsqueda de evento

Permite buscar eventos por título, descripción, localización o fecha.

Listado de eventos

Lista los eventos dados de alta por los gestores. En el listado público se podrá configurar si se muestran o no también los eventos pasados mientas que en el listado privado se mostrarán todos y se podrán ordenar por distintos criterios.

Visualización de eventos en calendario

Marca en un calendario los días en los que hay eventos y al hacer click en un día, muestra los eventos que se van a llevar a cabo en ese día concreto.

Visualización de estadísticas globales

Mostrará una gráfica con el progreso de registros en todos los eventos publicados, el total de plazas ofertadas y el porcentaje de plazas cubiertas.

Visualización de estadísticas de un evento

Mostrará una gráfica con el progreso de registros en un evento concreto, el total de plazas ofertadas para el evento y el porcentaje de plazas cubiertas.

Subsistema de gestión de registros

El subsistema de gestión de registros se encargará de las altas, bajas, consultas y modificaciones de los registros realizados por los visitantes del portal en cada evento.

Registro de asistente

Permitirá el registro de un visitante en un evento publicado en el portal. En caso de que el evento requiera de aprobación previa, el evento se almacenará con estado “Pendiente de revisión”, en caso contrario, se guardará con estado “Aprobado”.

Aprobar un registro

Permite a un gestor aprobar un registro en un evento publicado que requiera que los registros se revisen antes de ser aceptados. Al aprobar el registro, se enviará una notificación al correo electrónico del usuario que se registró indicando que se ha aceptado.

Rechazar un registro

Permite a un gestor rechazar un registro en un evento publicado que requiera que los registros se revisen antes de ser aceptados. Al rechazar el registro, se enviará una notificación al correo electrónico del usuario que se registro indicando que ha sido rechazada y un mensaje que introducirá el gestor.

Listado de registros

Lista todos los registros realizados en un evento, la opción en la que se han registrado y su estado. Al pinchar en un registro en el listado podremos ir a la opción de “Ver detalles de registro”.

Búsqueda de registro

Permite buscar registros por nombre del asistente, correo electrónico y estado del registro.

Exportación a Excel

Exporta el listado completo de registros para un evento en formato Excel.

Ver detalles de registro

Muestra los detalles de un registro completo con todos los datos introducidos durante el registro.

Imprimir registro

Imprime los detalles de un registro concreto.

Análisis funcional

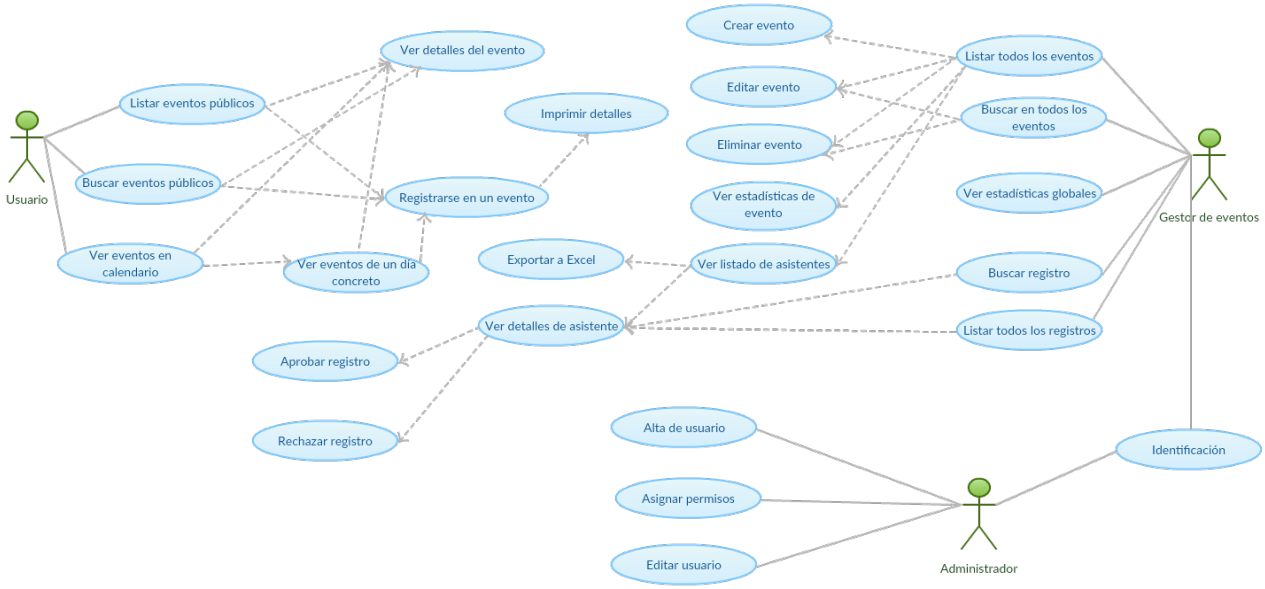
El análisis funcional de la aplicación se realizará utilizando una metodología de desarrollo ágil; la especificación de requisitos se ha realizado mediante historias de usuario. Aunque idealmente, estas historias las habrían de escribir los propios *stakeholders*, ante la inexistencia de participantes reales, en este caso las definiremos basándonos en el funcionamiento habitual de aplicaciones con el mismo propósito.

Además de detallar las historias de usuario, de cara a mostrar una visión general del sistema a implementar, también se presenta el diagrama de casos de uso.

Diagrama de casos de uso

Dentro del sistema de Event Registration se identifican tres roles diferentes:

- **Usuario (Guest)**: usuario anónimo, no identificado en el portal.
- **Gestor de eventos**: usuario identificado en el portal y con rol de “Gestor de eventos”.
- **Administrador**: usuario administrador del portal, tendrá permisos para realizar todas las acciones sobre el mismo.



Historias de usuario

Como usuario quiero ver el listado de eventos para poder saber cuales son los siguientes eventos disponibles.

Criterios de aceptación:

- Quiero ver primero los eventos que se van a realizar más pronto.
- Quiero poder navegar entre los resultados mediante páginas para poder ver todos los eventos.
- No debo poder ver los eventos que no están publicados.

Como usuario quiero buscar entre los eventos disponibles para poder listar los que se ajustan a mis necesidades.

Criterios de aceptación:

- Quiero poder buscar eventos por título, descripción, localización o fecha.
- No debo poder buscar sin rellenar alguno de los criterios.
- No debo poder buscar si no hay ningún evento disponible.
- En caso de que ningún evento se ajuste a mi búsqueda, se me debe informar de ello.

Como usuario quiero ver los eventos en un calendario para poder ver de un vistazo rápido los eventos que van a tener lugar y en qué fecha.

Criterios de aceptación:

- En caso de que el día sólo tenga un evento me ha de dirigir directamente al detalle.
- En caso de que el día tenga varios eventos, me muestre el listado de eventos de ese día.
- En caso de que no haya eventos un día concreto, no debe dirigirme a ningún sitio.

Como usuario quiero ver los detalles de un evento para comprobar si me interesa acudir a él y se ajusta a mi horario.

Criterios de aceptación:

- Quiero poder ver dónde va a tener lugar el evento y en qué momento.
- Quiero poder ver si el evento es de pago o gratuito.

- Quiero poder ver las plazas disponibles en cada horario.
- Quiero poder ver la descripción del evento, por ejemplo, quien lo organiza y las condiciones para la asistencia.

Como usuario quiero registrarme en un evento para poder acudir a él en el horario que mejor se ajuste a mi agenda.

Criterios de aceptación:

- Quiero poder registrarme en un evento si hay plazas libres.
- Quiero saber antes de registrarme si el registro requiere de aprobación.
- No debo poder registrarme en un evento si ya estoy registrado en él.
- En caso de que no haya plazas en el evento, se me debe informar antes de que intente registrarme.

Como usuario quiero imprimir la información de registro para poder guardar la información del registro.

Criterios de aceptación:

- Quiero poder imprimir los detalles del registro si se ha realizado correctamente.
- En caso de que falle el registro, no he de poder imprimir nada.

Como gestor de eventos quiero listar todos los eventos para poder ver todos los eventos que se han dado de alta en un sitio web.

Criterios de aceptación:

- Para cada evento, quiero ver su título, estado, creador y fecha de creación.
- Quiero poder ver todos los eventos, independientemente de su estado.
- Quiero ver primero los eventos que se han añadido hace menos tiempo.
- En caso de que haya muchos eventos, he de poder navegar entre ellos.
- En caso de que no haya ningún evento, se me debe informar de ello.

Como gestor de eventos quiero buscar entre todos los eventos para poder encontrar los que coincidan con ciertos criterios.

Criterios de aceptación:

- Quiero poder buscar eventos por título, descripción, localización, fecha o estado.
- No debo poder buscar sin rellenar alguno de los criterios.
- No debo poder buscar si no hay ningún evento creado.
- En caso de que ningún evento se ajuste a mi búsqueda, se me debe informar de ello.

Como gestor de eventos quiero ver estadísticas globales de creación de eventos y registros para poder ver el progreso en el registro de eventos.

Criterios de aceptación:

- Quiero poder ver estadísticas del último mes.
- Quiero poder navegar entre las estadísticas de otros periodos.

Como gestor de eventos quiero listar todos los registros para poder ver la información de todos los registros que se han producido en el portal.

Criterios de aceptación:

- Quiero poder listar los registros independientemente del evento al que pertenezcan.
- Quiero ver primero los registros que se han realizado hace menos tiempo.
- Quiero poder navegar entre todos los registros.
- En caso de que no haya ningún registro, se me debe informar de ello.

Como gestor de eventos quiero buscar entre todos los registros para poder encontrar usuarios registrados en función de ciertos criterios.

Criterios de aceptación:

- Quiero poder buscar registros por nombre, apellidos, correo electrónico o estado.

- No debo poder buscar sin rellenar alguno de los criterios.
- No debo poder buscar si no se ha realizado ningún registro.
- En caso de que ningún registro se ajuste a mi búsqueda, se me debe informar de ello.

Como gestor de eventos quiero crear eventos para poder publicarlos en el portal y que los usuarios puedan registrarse en ellos.

Criterios de aceptación:

- Quiero poder crear eventos como borrador para que pueda editarlos posteriormente sin que aparezcan todavía en el portal.
- Quiero poder incluir distintos horarios dentro de un mismo evento.
- No debo poder crear un evento sin rellenar su título, descripción, localización, fecha y hora de inicio.

Como gestor de eventos quiero editar los eventos para poder modificar su información y que esos cambios se reflejen en la información que ven los usuarios en el portal.

Criterios de aceptación:

- Quiero poder editar los eventos independientemente de su estado.
- En caso de que el evento este en estado de borrador, debo poder guardar los cambios en este mismo estado o pasarlo a estado publicado.
- En caso de que ya hubiera usuarios registrados al evento y se modifique su localización, fecha u hora, se debe enviar un correo a los usuarios indicando este cambio.
- No debo poder dejar en blanco la información referente al título del evento, descripción, localización, fecha y hora.

Como gestor de eventos quiero ver las estadísticas de un evento para poder evaluar el progreso de los registros.

Criterios de aceptación:

- Quiero poder sólo las estadísticas de los eventos publicados.

Como gestor de eventos quiero ver los usuarios registrados en un evento para poder saber cuantas personas se han registrado y quienes son.

Criterios de aceptación:

- Quiero poder listar sólo los registros en el evento que haya elegido.
- Quiero ver primero los registros que se han realizado hace menos tiempo.
- Quiero poder navegar entre todos los registros.
- En caso de que no haya ningún registro, se me debe informar de ello.

Como gestor de eventos quiero ver la información de un registro para poder saber quien se ha registrado para poder aprobar o rechazar su registro y cómo contactar con dicha persona.

Criterios de aceptación:

- Quiero poder listar sólo los registros en el evento que haya elegido.
- Quiero ver primero los registros que se han realizado hace menos tiempo.

Como gestor de eventos quiero rechazar un registro para que no puedan asistir aquellos que no cumplan los requisitos o que incluyan datos falsos.

Criterios de aceptación:

- Quiero poder rechazar sólo los registros que estén en estado “Pendiente”.
- Quiero que el usuario reciba un correo electrónico indicando que su registro ha sido aprobado.
- No debo poder rechazar un registro que ya esté aprobado.

Como gestor de eventos quiero exportar el listado de registros de un evento a Excel para poder tratar o consultar el listado fuera del portal.

Criterios de aceptación:

- Quiero poder exportar sólo los registros de un evento concreto.

- Quiero que el listado incluya todos los datos disponibles sobre el registro.
- No debo poder exportar si no hay ningún registro.

Como administrador quiero dar de alta usuarios para poder asignarle permisos en el portal y delegar en él ciertas tareas de gestión.

Criterios de aceptación:

- No debo poder dar de alta un usuario sin introducir su nombre, correo electrónico y nombre de usuario.

Como administrador quiero editar un usuario para poder actualizar su información o cambiar su contraseña de acceso.

Criterios de aceptación:

- No debo poder editar un usuario si borro su nombre, correo electrónico o nombre de usuario.

Como administrador quiero asignar permisos a un usuario para poder limitar lo que cada usuario puede gestionar.

Criterios de aceptación:

- Quiero poder definir grupos de permisos (roles) y asignar éstos a cada usuario.

Estados

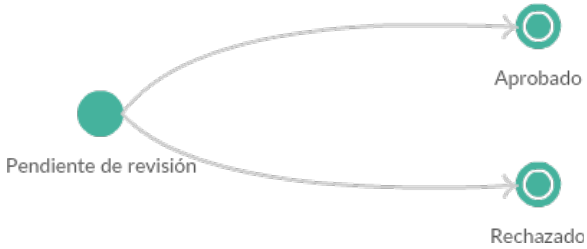
Para poder implementar las funcionalidades descritas anteriormente necesitaremos poder establecer distintos estados para algunos de nuestros modelos, concretamente para los eventos y registros.

En el caso de los eventos, podrán encontrarse en los siguientes estados:

- **Borrador**: el evento se ha creado pero todavía no se ha completado, por lo que todavía no deben poder verlo los usuarios.

- **Publicado**: el evento se ha completado y los usuarios ya pueden verlo y registrarse en él.

Por otra parte, los registros pueden encontrarse en alguno de los siguientes estados:



- **Pendiente de revisión**: el usuario ha rellenado el formulario y los datos se han introducido en el sistema.
- **Aprobado**: un gestor ha aprobado el registro o éste ha sido aprobado automáticamente en el caso de eventos que no requieran de aprobación previa.
- **Rechazado**: un gestor ha rechazado el registro introduciendo un motivo para el rechazo.

Tecnologías y arquitectura

Para desarrollar esta aplicación utilizaremos Liferay Portal y la tecnología de Portlets, de esta forma tendremos ya resueltas muchas de las funcionalidades inherentes a portales en línea, como todo el subsistema de gestión de usuarios y permisos. Además, nos proporciona toda una serie de componentes y frameworks de desarrollo.

Para instalar nuestra infraestructura básica de portal Liferay utilizaremos los siguientes componentes:

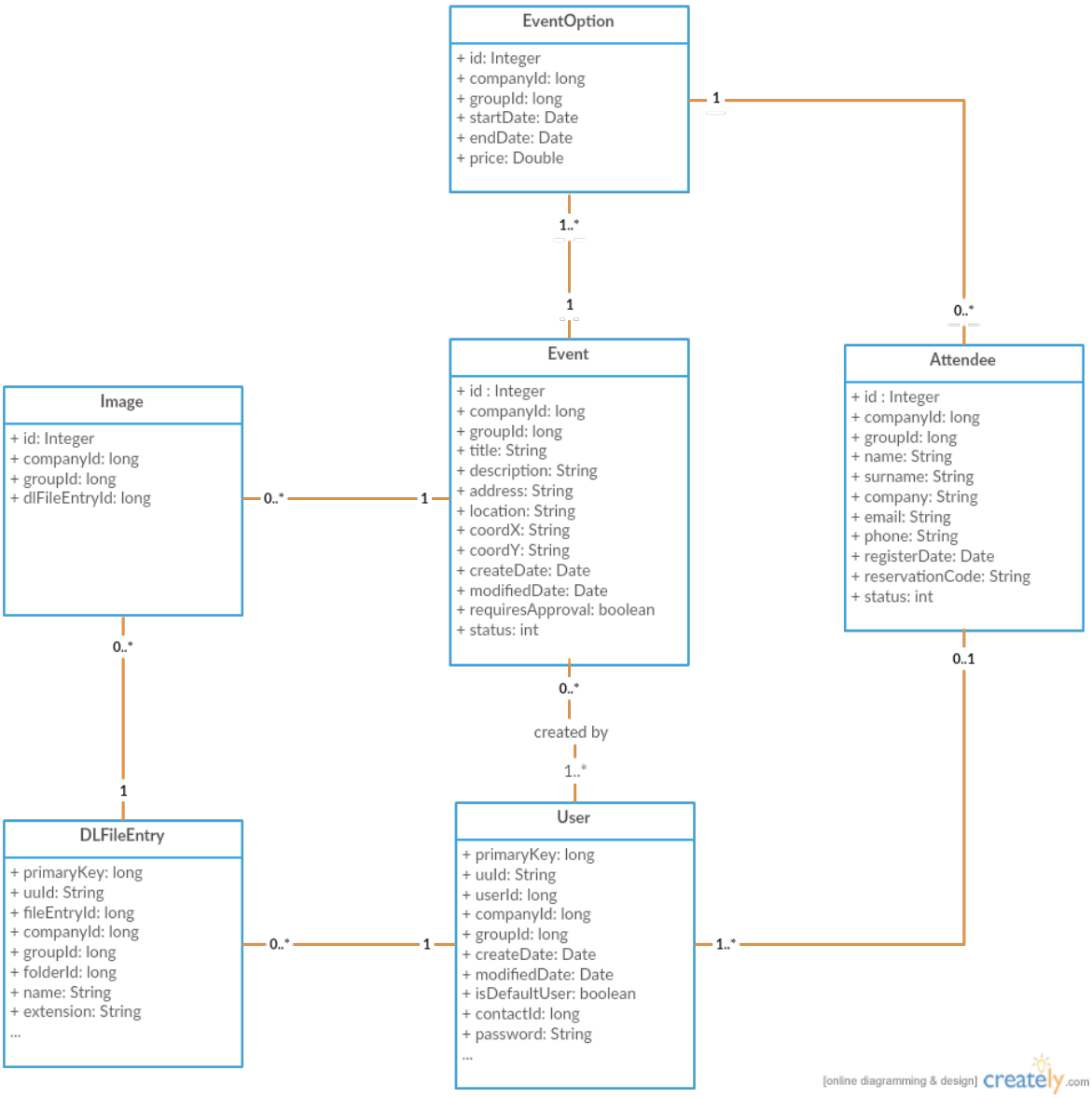
- Apache Tomcat como servidor de aplicaciones
- Liferay Portal 6.2 Community Edition como contenedor de portlets. En estos momentos, se está desarrollando la versión 7 de Liferay, pero todavía se encuentra en fase beta, por lo que elegimos la última versión estable.
- Oracle Database 11g Express Edition como base de datos.

Para el desarrollo de los portlets, se utilizarán las siguientes tecnologías:

- Eclipse y el plugin Liferay IDE para el desarrollo de componentes para Liferay Portal.
- Liferay Plugins SDK y Ant
- Git como herramienta de control de versiones
- JSR 286 y su implementación en Liferay, Portlet MVC
- Service Builder para la construcción de modelos y sus servicios (Spring y Hibernate).
- JSP y los taglibs que nos ofrece Liferay (liferay-ui, liferay-portlet, etc).
- La librería Java Apache POI para implementar la exportación a Excel.
- Bootstrap y jQuery para la presentación de las vistas.
- La librería JavaScript Highcharts para generar las gráficas de estadísticas.

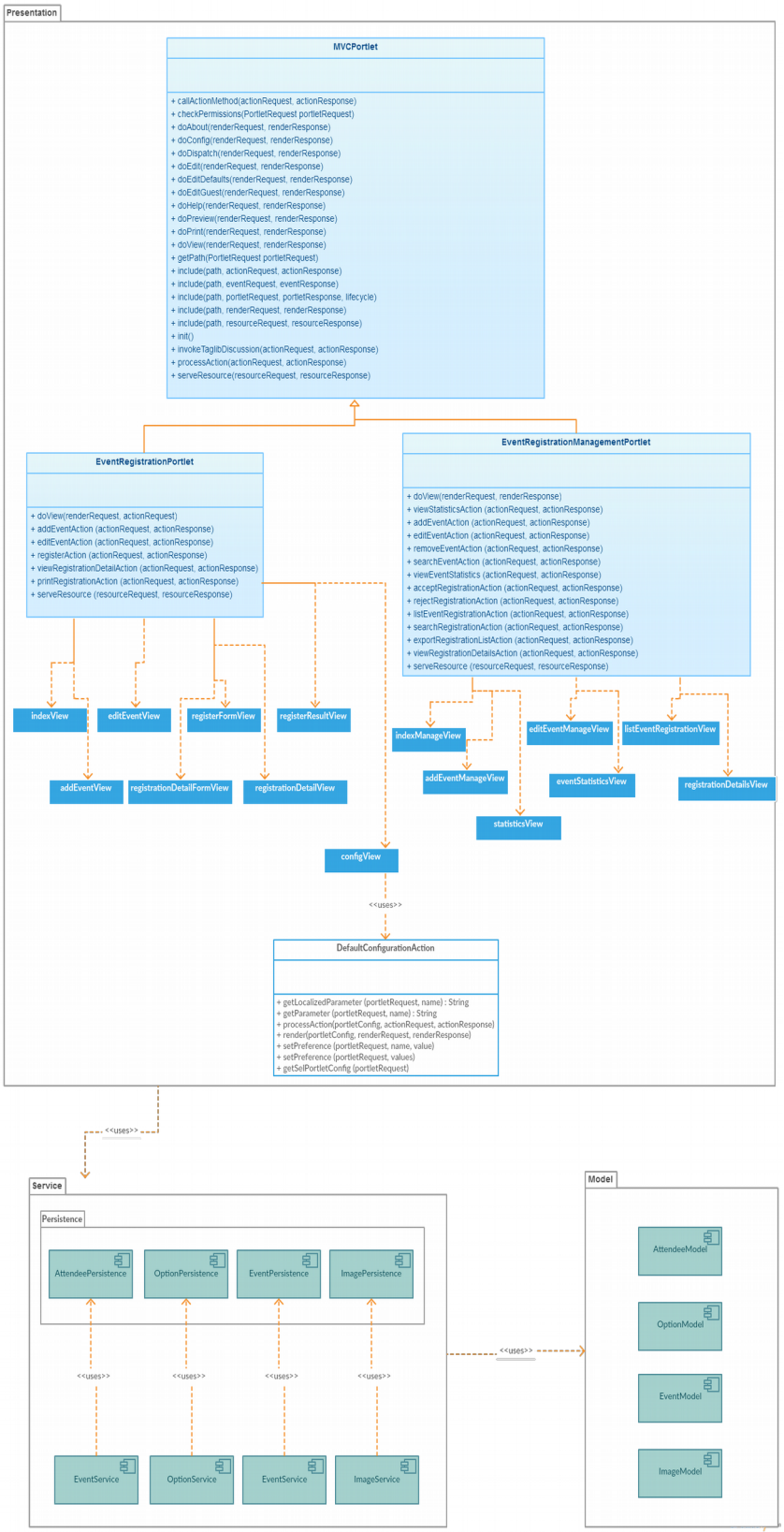
Diagrama de entidades y clases

El diagrama de entidades de nuestra aplicación quedaría de la siguiente manera:



Las entidades DLFileEntry y User son propias del portal, pero se relacionarán con las tablas que generemos para nuestros portlets. Todas las tablas tienen incorporados los atributos *companyId* y *groupId*, esto se debe a que Liferay es multi-instancia (companyId) y multi-site (groupId) y dado que es previsible que cada sitio web creado en el portal gestione sus propios eventos tal y como ocurre con contenidos y documentos, es necesario que incluyamos esta información en nuestros modelos.

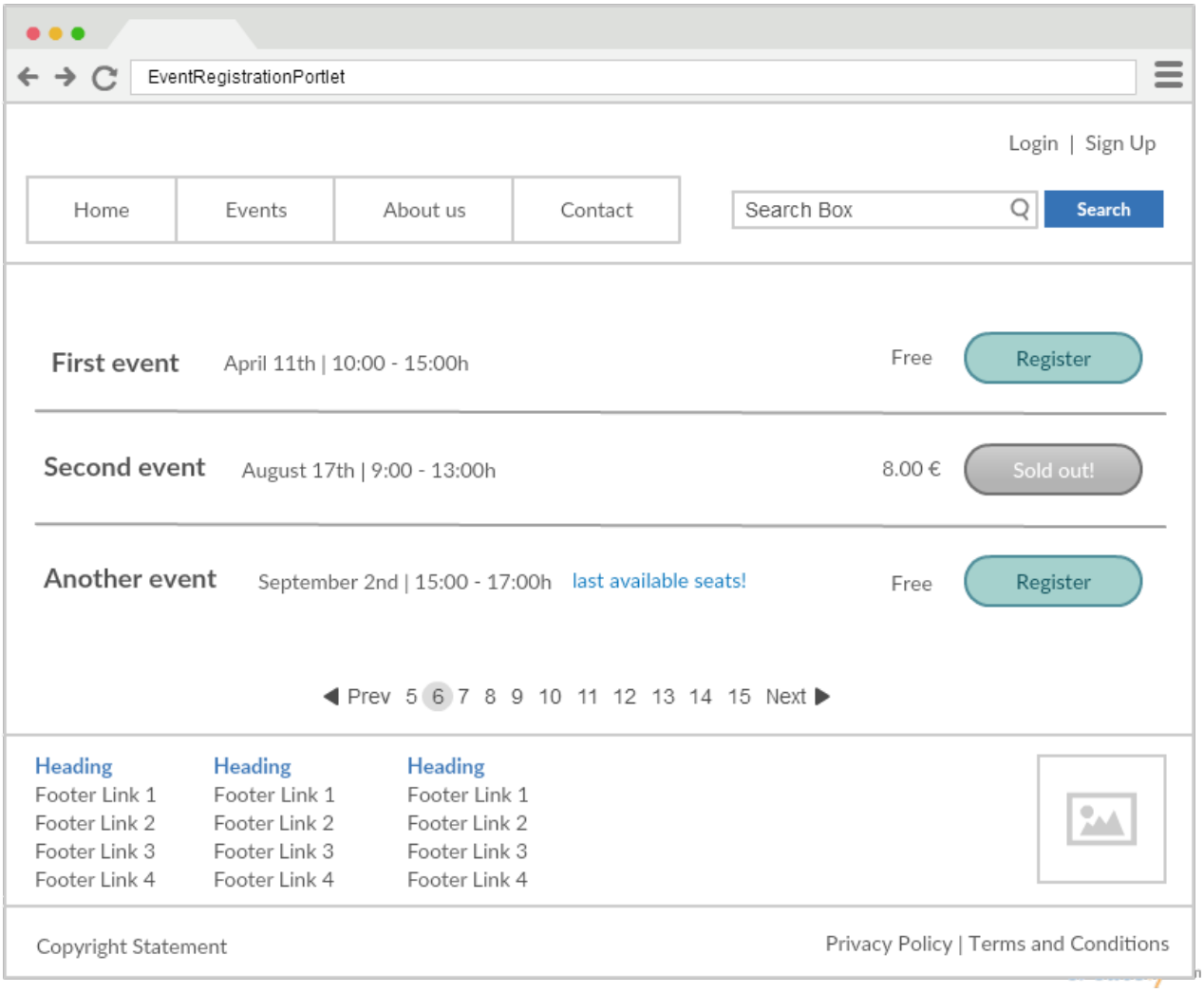
A continuación se presenta el diagrama de clases de la aplicación. No se entra en detalle en las clases de servicio, modelo y persistencia ya que generaría un diagrama demasiado grande y complejo donde se perdería la visión global de la implementación:


Presentation
MVCPortlet
+ callActionMethod(actionRequest, actionResponse)
+ checkPermissions(PortletRequest portletRequest)
+ doAbout(renderRequest, renderResponse)
+ doConfig(renderRequest, renderResponse)
+ doDispatch(renderRequest, renderResponse)
+ doEdit(renderRequest, renderResponse)
+ doEditDefaults(renderRequest, renderResponse)
+ doEditGuest(renderRequest, renderResponse)
+ doHelp(renderRequest, renderResponse)
+ doPreview(renderRequest, renderResponse)
+ doPrint(renderRequest, renderResponse)
+ doView(renderRequest, renderResponse)
+ getPath(PortletRequest portletRequest)
+ include(path, actionRequest, actionResponse)
+ include(path, eventRequest, eventResponse)
+ include(path, portletRequest, portletResponse, lifecycle)
+ include(path, renderRequest, renderResponse)
+ include(path, resourceRequest, resourceResponse)
+ init()
+ invokeTaglibDiscussion(actionRequest, actionResponse)
+ processAction(actionRequest, actionResponse)
+ serveResource(resourceRequest, resourceResponse)
EventRegistrationPortlet
+ doView(renderRequest, actionRequest)
+ addEventAction (actionRequest, actionResponse)
+ editEventAction (actionRequest, actionResponse)
+ registerAction (actionRequest, actionResponse)
+ viewRegistrationDetailAction (actionRequest, actionResponse)
+ printRegistrationAction (actionRequest, actionResponse)
+ serveResource (resourceRequest, resourceResponse)
EventRegistrationManagementPortlet
+ doView(renderRequest, renderResponse)
+ viewStatisticsAction (actionRequest, actionResponse)
+ addEventAction (actionRequest, actionResponse)
+ editEventAction (actionRequest, actionResponse)
+ removeEventAction (actionRequest, actionResponse)
+ searchEventAction (actionRequest, actionResponse)
+ viewEventStatistics (actionRequest, actionResponse)
+ acceptRegistrationAction (actionRequest, actionResponse)
+ rejectRegistrationAction (actionRequest, actionResponse)
+ listEventRegistrationAction (actionRequest, actionResponse)
+ searchRegistrationAction (actionRequest, actionResponse)
+ exportRegistrationListAction (actionRequest, actionResponse)
+ viewRegistrationDetailsAction (actionRequest, actionResponse)
+ serveResource (resourceRequest, resourceResponse)
indexView
editEventView
registerFormView
registerResultView
addEventView
registrationDetailFormView
registrationDetailView
indexManageView
addEventManageView
editEventManageView
eventStatisticsView
statisticsView
listEventRegistrationView
registrationDetailsView
configView
<<uses>>
DefaultConfigurationAction
+ getLocalizedParameter (portletRequest, name) : String
+ getParameter (portletRequest, name) : String
+ processAction(portletConfig, actionRequest, actionResponse)
+ render(portletConfig, renderRequest, renderResponse)
+ setPreference (portletRequest, name, value)
+ setPreference (portletRequest, values)
+ getSelPortletConfig (portletRequest)
<<uses>>
Service
Persistence
AttendeePersistence
OptionPersistence
EventPersistence
ImagePersistence
<<uses>>
<<uses>>
<<uses>>
<<uses>>
EventService
OptionService
EventService
ImageService
<<uses>>
Model
AttendeeModel
OptionModel
EventModel
ImageModel

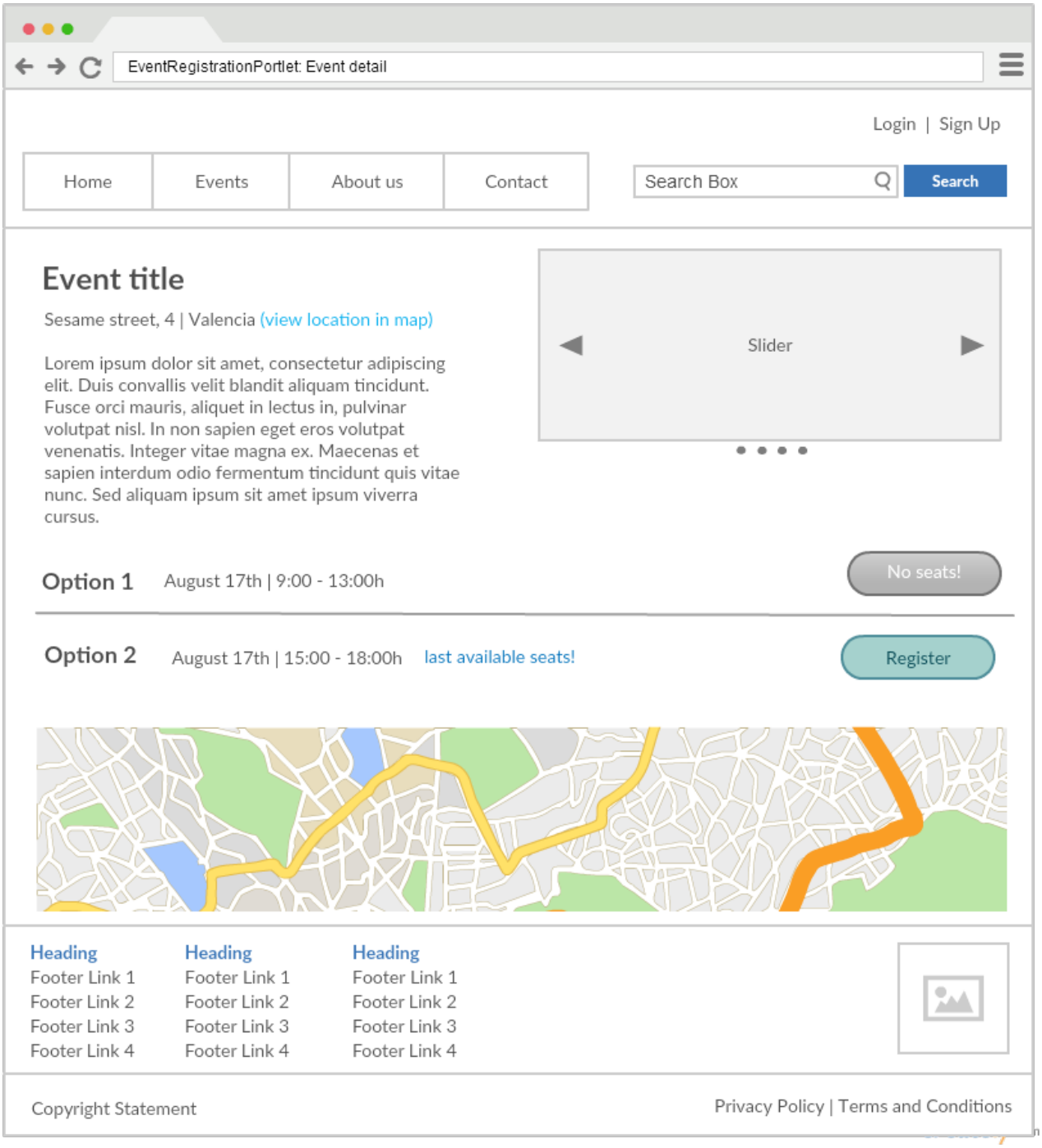
Prototipo de pantallas principales

A continuación se presentan prototipos para algunas de las pantallas principales de la aplicación.

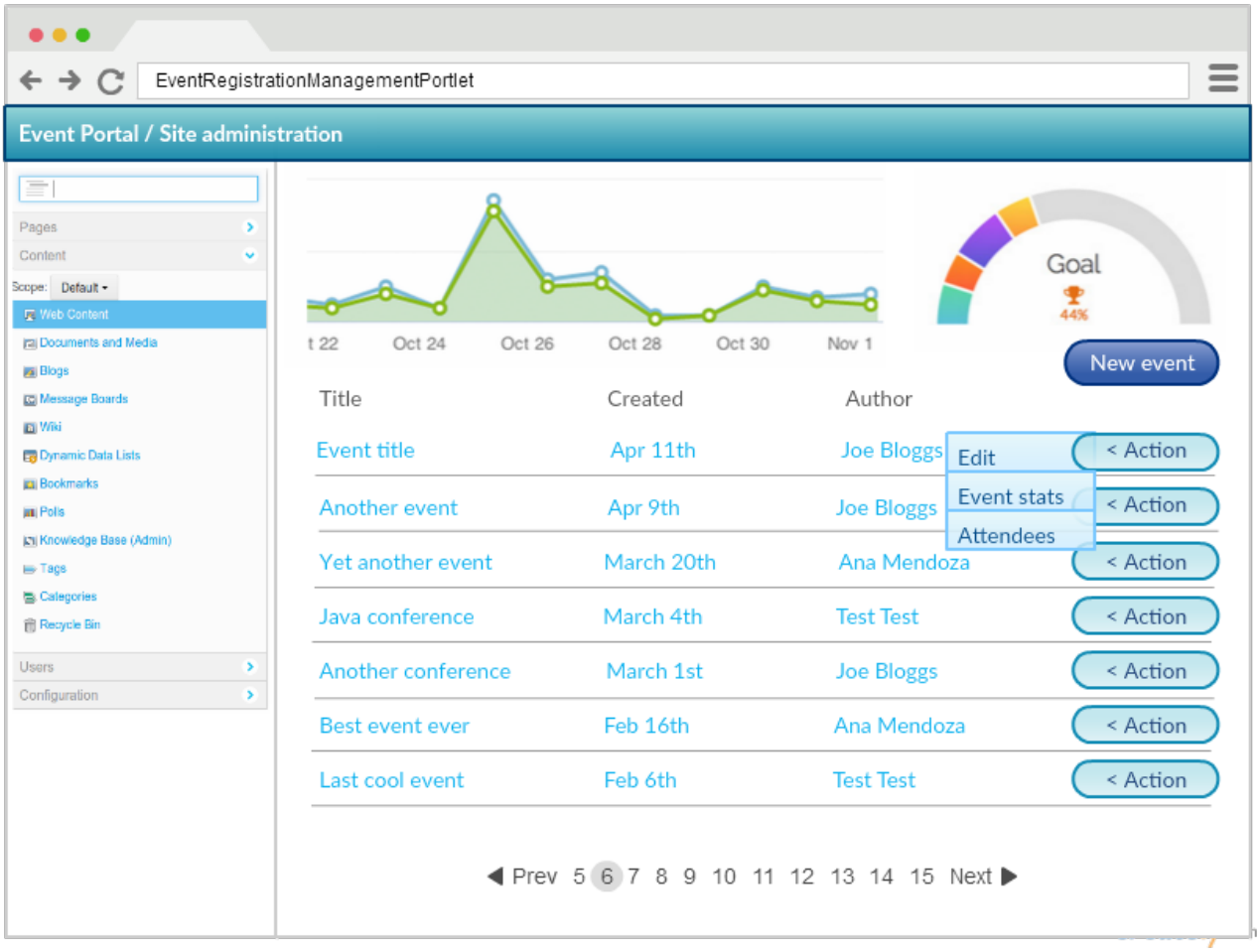
Listado de eventos, pantalla principal del portlet público (EventRegistrationPortlet):



Detalle de evento del portlet público (EventRegistrationPortlet):



Pantalla principal del portlet de gestión de eventos (EventRegistrationManagementPortlet):



Planificación

Se establece una planificación de tareas basada en hitos marcados por las fechas de entrega del TFG.

Hito	Fecha	Descripción
Plan de trabajo: PAC1	14/03/2016	Definición del proyecto, toma de requisitos, definición de la arquitectura y diseño de diagramas de entidad y clase.
Análisis e instalación: PAC2	14/04/2016	Definición de las historias de usuario, diseño de pantallas a implementar, instalación del software necesario y creación inicial de la estructura de proyecto con sus entidades.
Implementación y documentación: PAC3	31/05/2016	Implementación del proyecto completo, realización de pruebas funcionales, documentación técnica y manual de usuario.
Memoria y presentación	24/06/2016	Realización de la memoria y la presentación del proyecto.

Implementación del sistema

Para implementar las funcionalidades que ha de tener la aplicación, he desarrollado dos portlets para Liferay Portal:

- Portlet público (*Event Registration Portlet*): contiene todas las funcionalidades públicas como el listado de eventos y el formulario de registro y consulta.
- Portlet de panel de control (*Event Registration Management Portlet*): contendrá las funcionalidades de mantenimiento de eventos y reservas por parte de los gestores de eventos.

Ambos portlets se desarrollarán bajo un mismo proyecto, por lo que resultarán en un solo archivo distribuible (war).

La lógica de estos portlets se desarrollará extendiendo de la clase *MVCPortlet*, por lo que tendremos dos clases de manejo de portlets, una para cada uno de ellos: *EventRegistrationPortlet* y *EventRegistrationManagementPortlet*.

Se generará un fichero basado en Hibernate, *service.xml*, donde se definirán los modelos de nuestro sistema. A partir de este fichero se generarán toda una serie de clases Java para el acceso local y remoto a los métodos de la capa de negocio e integración de nuestra aplicación.

Instalación del software necesario

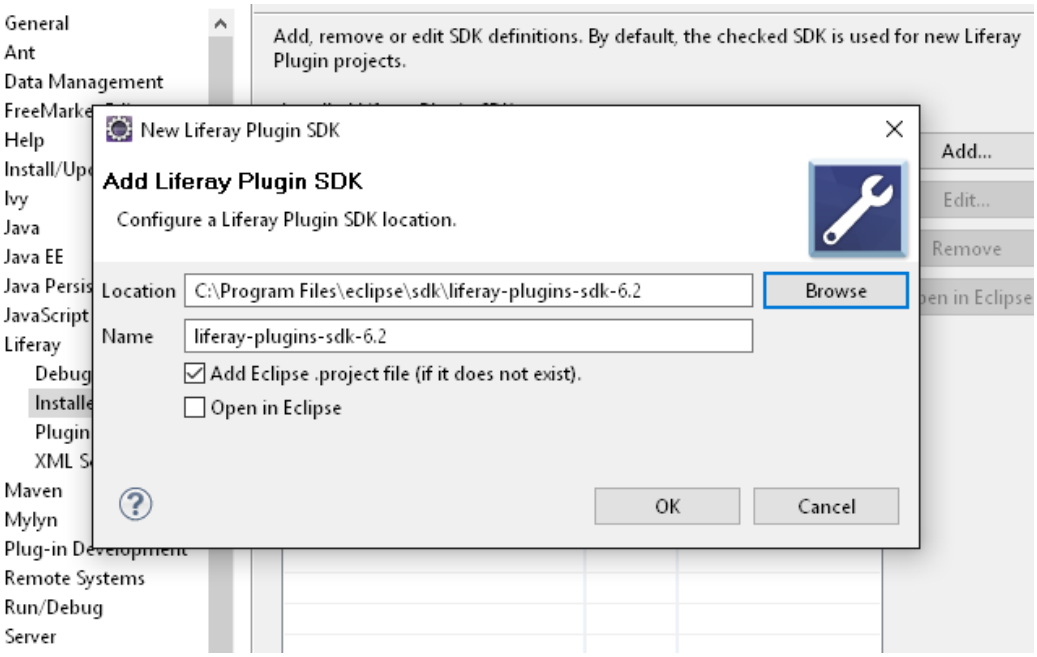
Durante la implementación del proyecto se ha publicado la primera versión estable de Liferay 7, pero el plugin para eclipse para la versión 7 todavía no tiene disponible una versión estable y su instalación da algunos problemas, por lo que se desarrollará sobre la versión 6.2 CE. Aunque inicialmente se propuso instalar el portal sobre una base de datos Oracle Express Edition, finalmente he utilizado una base de datos postgreSQL dado que ya disponia de un servidor de este tipo instalado en local. En cualquier caso, la elección de SGDB no condiciona el desarrollo del proyecto ni su instalación, ya que los portlets disponen de una capa de abstracción con Hibernate que permite que los desarrollos funcionen de forma independiente al SGDB escogido.

Entorno de desarrollo: Liferay IDE

Dado que vamos a desarrollar sobre la versión Community de Liferay, instalaremos Liferay IDE sobre eclipse ya que Liferay Developer Studio requiere que haya configurado como mínimo un servidor de la versión Enterprise para funcionar. Aunque podríamos configurar un servidor EE con una versión de prueba que puede descargarse desde la web www.liferay.com, elegimos instalar el plugin de eclipse para saltar este paso.

Descargamos la versión *Eclipse IDE for Java EE Developers* para empezar con una instalación limpia y sobre ella, instalamos Liferay IDE desde la opción “Help > Install new software...” e introducimos la URL de descarga de la versión estable del IDE que podemos obtener desde la página de Liferay.

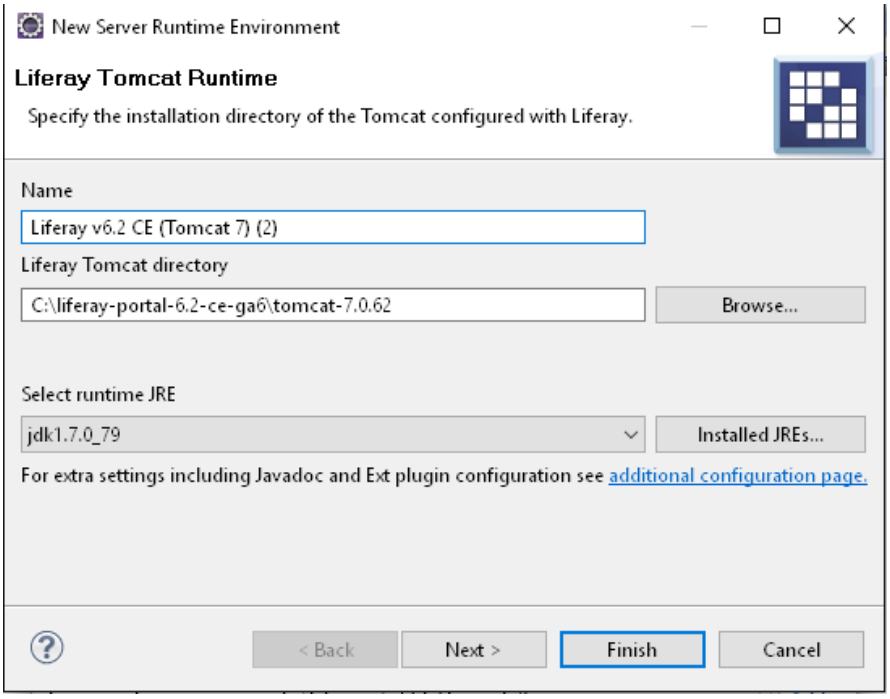
Necesitaremos también el Liferay SDK para desarrollar nuestros portlets; lo descargamos desde la misma página, lo copiamos a una carpeta de nuestra instalación de eclipse y lo configuramos desde eclipse en la opción de “Liferay > Installed Plugin SDKs > Add...”



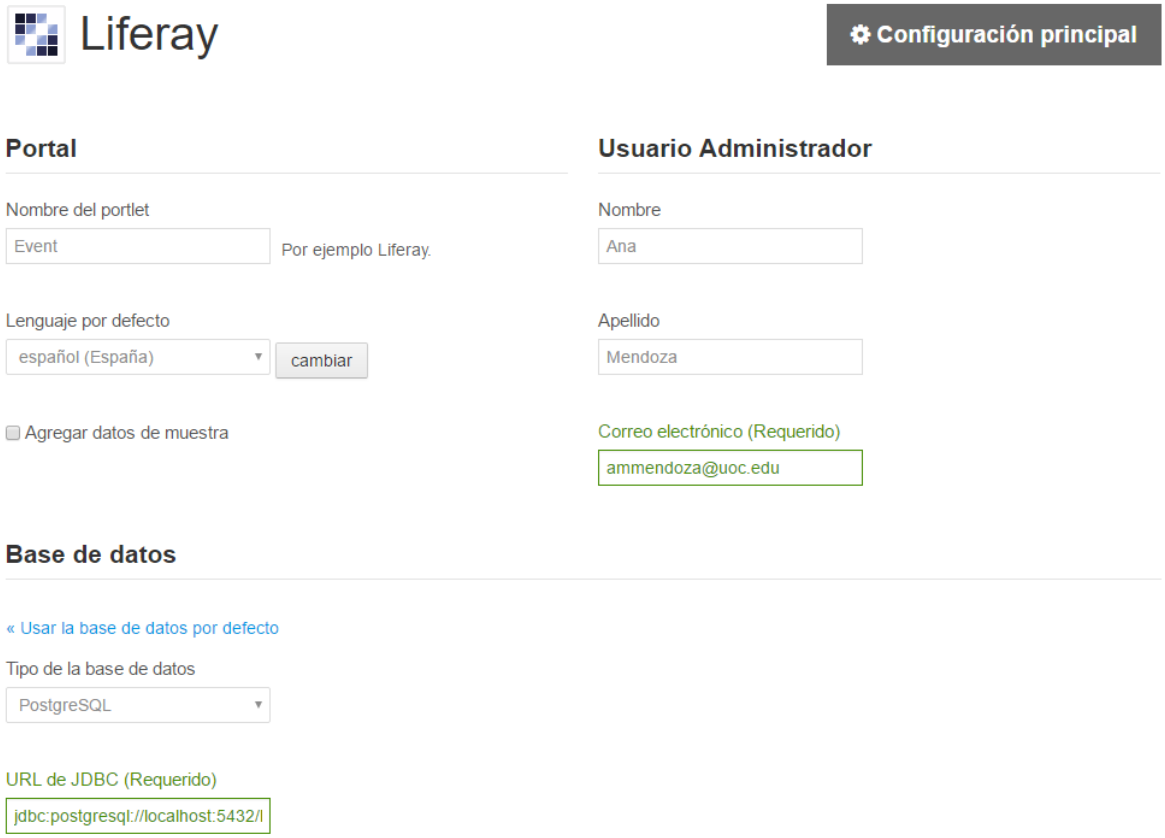
Liferay Portal

Desde la web de Liferay, descargamos Liferay Portal 6.2 CE GA6. Podemos descargar el software ya empaquetado con el servidor de aplicaciones Java, que podemos elegir entre Tomcat, Glassfish, Jboss o Jetty; elegimos Tomcat por su sencillez y el hecho de que se trata de una instalación en un entorno local. Para instalarlo, simplemente copiamos el bundle en nuestro equipo local, preferiblemente en una ruta corta, por ejemplo: c:/, ya que el copiado puede dar problemas causados por rutas demasiado largas en Windows.

También debemos configurar el servidor de aplicaciones en eclipse para poder desplegar automáticamente los cambios durante el desarrollo e iniciar el servidor en modo debug, para ello, añadimos un nuevo servidor y elegimos de la lista que nos aparece la versión del servidor de aplicaciones del bundle que hemos descomprimido y su ruta:



Liferay Portal puede funcionar con diversos sistemas de gestión de bases de datos, incluso puede funcionar con un sistema de base de datos en fichero, llamado Hypersonic, que es la opción por defecto, pero para que el rendimiento sea adecuado durante el desarrollo, configuraremos el portal para que funcione contra una base de datos postgreSQL que creamos con pgAdminII. Iniciamos el servidor desde eclipse y accedemos por navegador web a http://localhost:8080 donde se nos muestra el configurador inicial. El bundle de Liferay + Tomcat ya tiene por defecto incluida la librería de conexión con postgreSQL, por lo que podemos introducir directamente la información de la base de datos y el usuario que hemos creado previamente:



Desmarcamos la opción de “Agregar datos de muestra” para evitar que se almacenen datos innecesarios en nuestra base de datos.

Una vez hacemos click en el botón de instalación, Liferay crea toda la estructura de tablas, introduce los datos necesarios y verifica que sean correctos, podemos ver el progreso en la consola de eclipse:

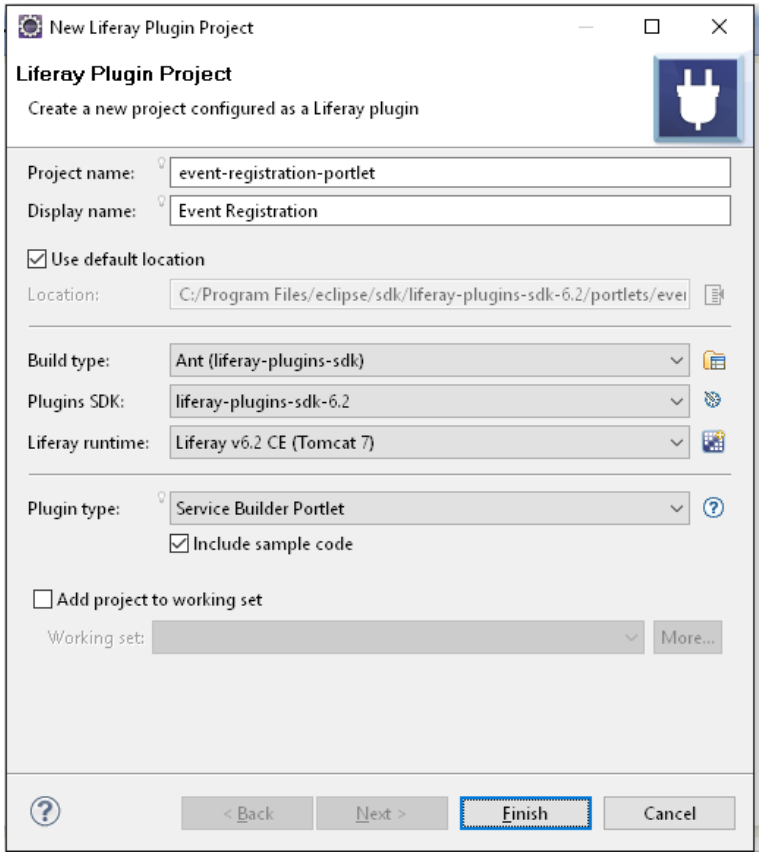
```
abr 11, 2016 7:27:38 PM org.apache.catalina.startup.Catalina start
INFORMACIÓN: Server startup in 240127 ms
19:32:13,001 INFO  [http-bio-8080-exec-10][DialectDetector:71] Determine dialect for PostgreSQL 9
19:32:13,011 INFO  [http-bio-8080-exec-10][DialectDetector:136] Found dialect org.hibernate.dialect.PostgreSQLDialect
Starting Liferay Portal Community Edition 6.2 CE GA6 (Newton / Build 6205 / January 6, 2016)
19:32:18,385 INFO  [http-bio-8080-exec-10][StartupAction:97] There are no patches installed
19:32:18,408 ERROR [http-bio-8080-exec-10][JDBCExceptionReporter:82] ERROR: no existe la relación «lock_»_  Position: 414 [Sanitized]
19:32:18,413 WARN  [http-bio-8080-exec-10][StartupAction:147] Unable to clear locks because Lock table does not exist
19:32:18,420 WARN  [http-bio-8080-exec-10][ReleaseLocalServiceImpl:171] ERROR: no existe la relación «release_»_  Position: 25 [Sanitized]
19:32:18,423 INFO  [http-bio-8080-exec-10][ReleaseLocalServiceImpl:84] Create tables and populate with default data
```

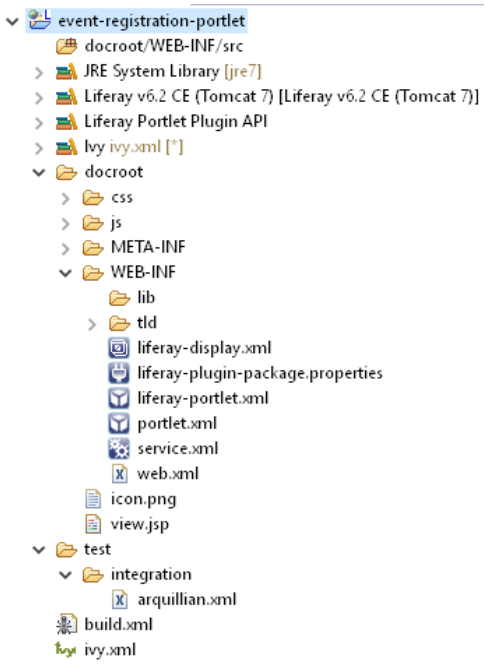
Finalmente, se nos presenta la pantalla de confirmación indicando que se ha instalado correctamente el portal y que las opciones de configuración que hemos introducido en la instalación se han guardado en el archivo *portal-setup-wizard.properties*. Muchas de características de Liferay se pueden ajustar mediante propiedades en ficheros de este tipo, que se pueden colocar tanto en

la carpeta principal de Liferay como dentro de la carpeta webapps\ROOT\WEB-INF\classes dentro de la carpeta de Tomcat.

Creación del proyecto

Una vez hemos configurado el entorno de desarrollo completo e instalado el portal ya podemos crear nuestro proyecto de plugin desde eclipse. Lo hacemos desde la opción “New > Liferay Plugin Project” del explorador de proyectos y creamos un plugin de tipo “Service Builder Portlet”:



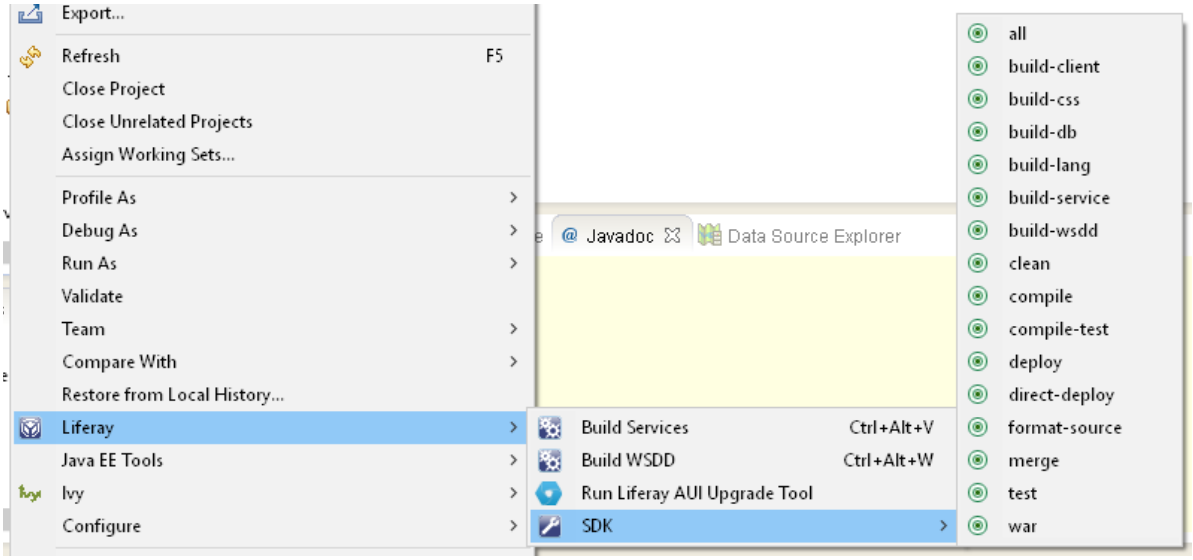


Como podemos comprobar una vez termina de crear el proyecto, el IDE crea toda la estructura del proyecto, incluyendo los ficheros de configuración y las tareas Ant de compilación y despliegue. Si abrimos alguno de los ficheros de configuración del portlet como, por ejemplo, el fichero portlet.xml veremos que se han incluido toda una serie de valores de muestra que tendremos que modificar durante la implementación del proyecto. Por ejemplo, por defecto aparece la siguiente opción de configuración:

<portlet-class>com.liferay.util.bridges.mvc.MVCPortlet</portlet-class>

Esta opción establece que la clase encargada de gestionar el portlet es MVCPortlet, la clase genérica de gestión de *Spring Portlet MVC*, sin embargo, para implementar nuestro portlet necesitaremos implementar la lógica de la vista principal y toda una serie de acciones, por lo que tendremos que crear una clase java que extienda de ésta y cambiar este valor de la configuración del portlet.

Al tratarse de un plugin de tipo “Service Builder”, si hacemos click derecho en el proyecto, podemos ver las siguientes tareas:

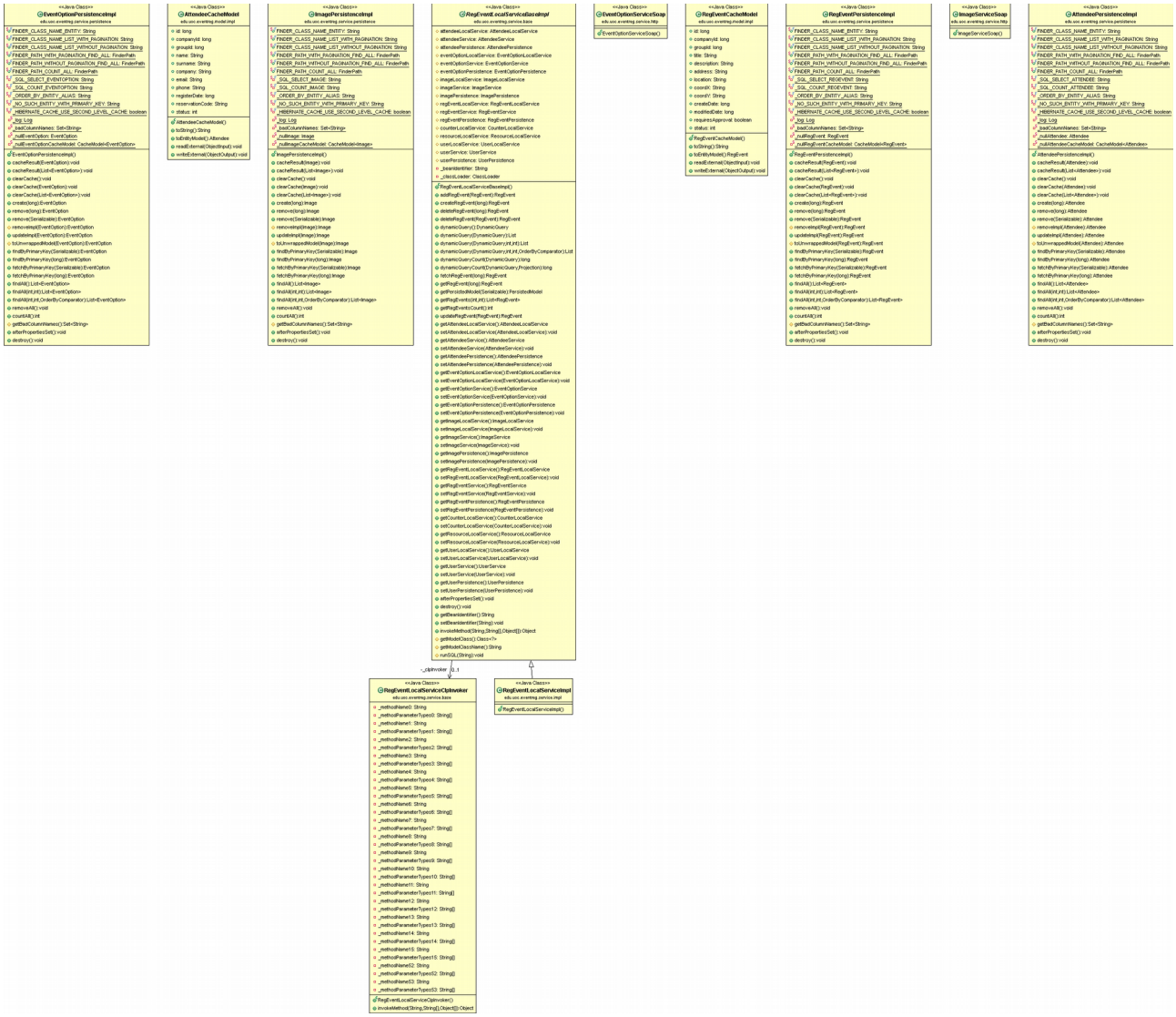


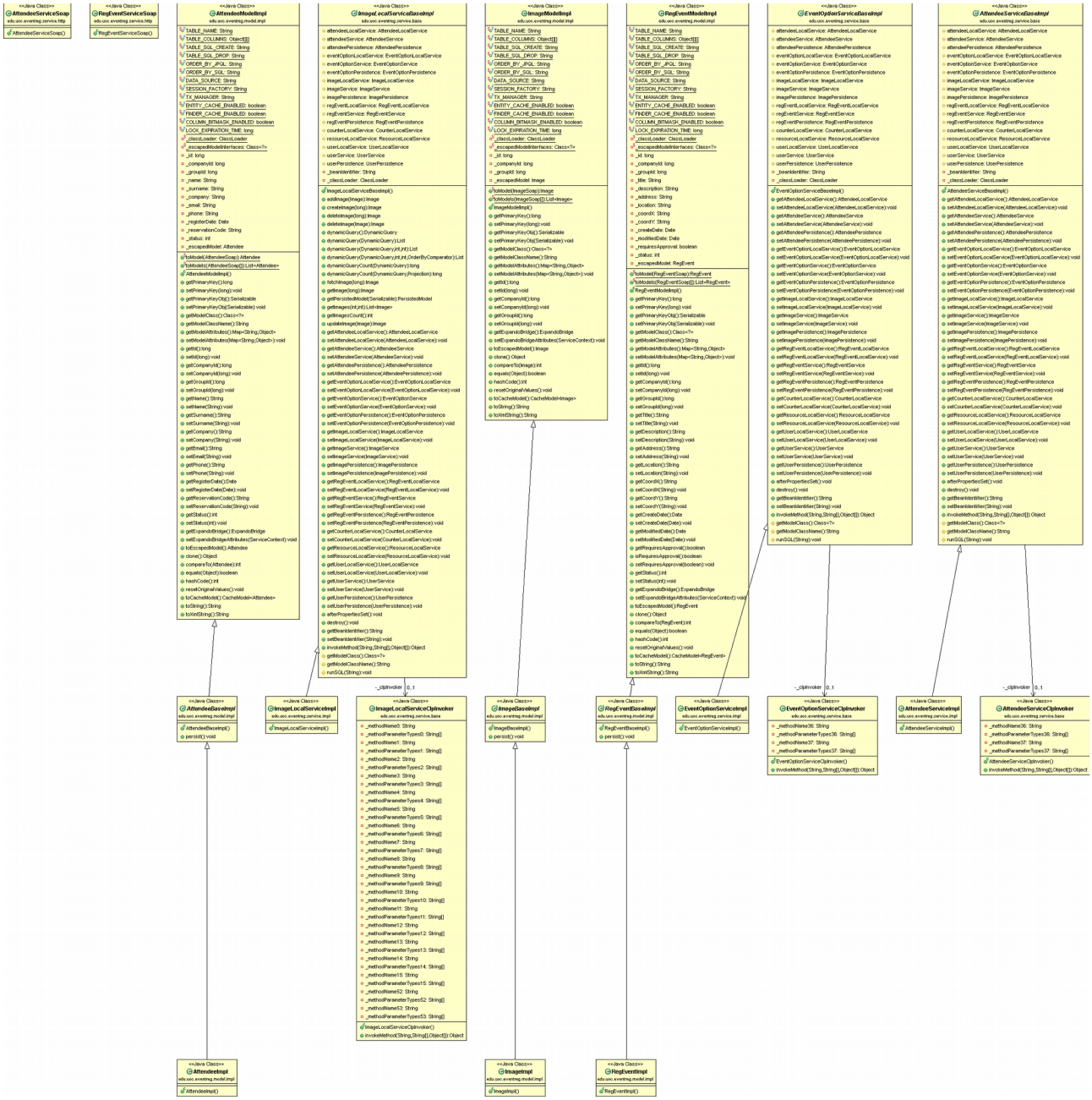
Al crear el proyecto ya hemos creado el primer portlet, “Event Registration Portlet” que tal y como definimos en la primera PAC, incluirá las funcionalidades accesibles desde el frontend. Además de este, creamos un segundo portlet “Event Registration Management Portlet” que será el portlet de gestión de eventos accesible sólo desde el backoffice del portal, haciendo click derecho en el proyecto y eligiendo “New > Liferay Portlet”. Se nos presenta un wizard de creación de portlet que completamos indicando las características del portet a crear como, por ejemplo, cual será su clase manejadora. Una vez completado, el wizard añade la configuración necesaria en el proyecto, pero podremos modificarla posteriormente.

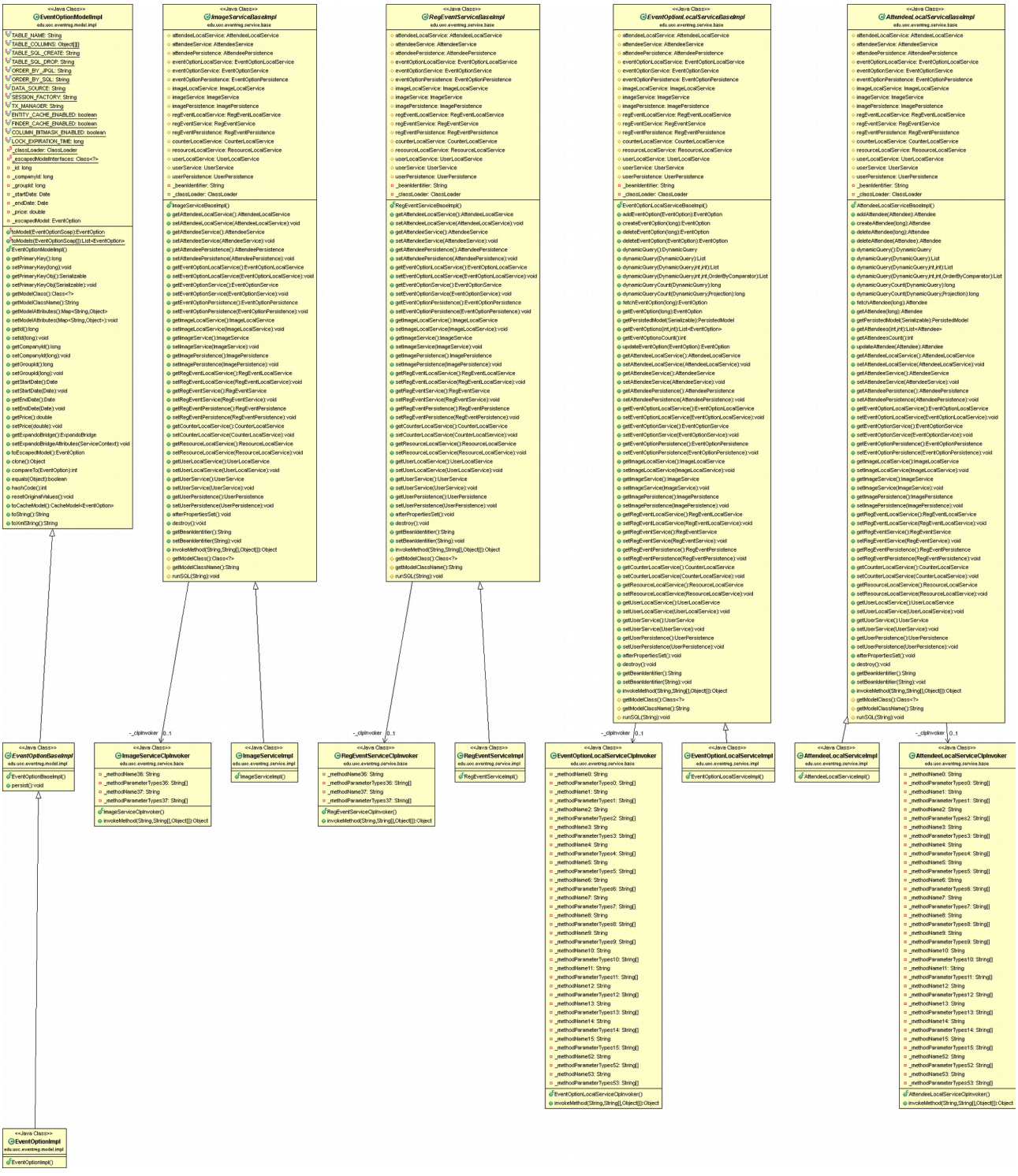
Generación automática del código

Para reflejar nuestro modelo en el proyecto, editaremos el fichero docroot/WEB-INF/service.xml e introduciremos la definición de nuestras entidades en formato XML. Cada una de las entidades corresponde con una tabla en base de datos, cuyo nombre coincide con el nombre de la entidad al que se le incluye delante un prefijo, común a todo el proyecto y que se define en el elemento “namespace” del fichero service.xml. Una vez hecho esto, procedemos a ejecutar la tarea “Build Services”.

En este punto, generamos el diagrama UML del proyecto, que en estos momentos queda de la siguiente manera:


EventOptionPersistenceImpl
AttendeeCacheModel
ImagePersistenceImpl
RegEventLocalServiceBaseImpl
EventOptionServiceSoap
RegEventCacheModel
RegEventPersistenceImpl
ImageServiceSoap
AttendeePersistenceImpl
RegEventLocalServiceClpInvoker
RegEventLocalServiceImpl





Como se puede observar, se han generado todas las clases de los modelos introducidos en el service.xml, las clases de servicio local, remoto y todas las clases de persistencia. Durante la implementación del proyecto necesitaremos ampliar los métodos disponibles desde las clases de servicio modificando las

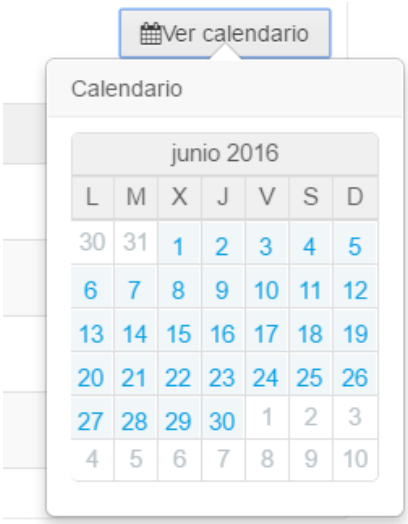
clases de implementación (**Impl*), pero ya disponemos de un punto de partida que nos proporciona muchas de las funciones que necesitaremos.

Decisiones de implementación

Aunque inicialmente pensaba incluir las imágenes de los eventos como un carrusel de imágenes, el componente de carrusel de AlloyUI incluido en la versión 6.1 de Liferay no es responsive, por lo que se ha escogido mostrar, en su lugar, un listado a modo de galería de imágenes.

Funcionalidades no implementadas

Debido al tiempo disponible para la implementación del proyecto, no ha sido posible implementar una de las funcionalidades que mostraba los eventos en un calendario y permitía ver los eventos que ocurrían en un día concreto. Al iniciar la implementación de esta funcionalidad me dí cuenta de que el comportamiento del componente de calendario no era el esperado, ya que ha variado desde la versión 6.1 de Liferay y ya no provee mecanismos de navegación entre meses, por lo que sería necesario que también implementara enlaces para poder visualizar todos los meses disponibles.



Por otra parte, aunque sí que se ha implementado la búsqueda de eventos y permite buscar por distintos campos, no ha sido posible habilitar un campo para buscar por la fecha en la que tiene lugar el evento; llegué a implementar esta opción, pero al añadir un campo de tipo fecha al buscador siempre se rellenaba con la fecha actual, lo cual hacía difícil distinguir si se trataba de una fecha introducida por el usuario o la fecha por defecto del campo.

Aunque no se trata de una funcionalidad, en la planificación se había incluido la realización de un manual de usuario. Finalmente no ha sido posible, ya que he necesitado la totalidad del tiempo para la implementación del proyecto.

Funcionalidades adicionales

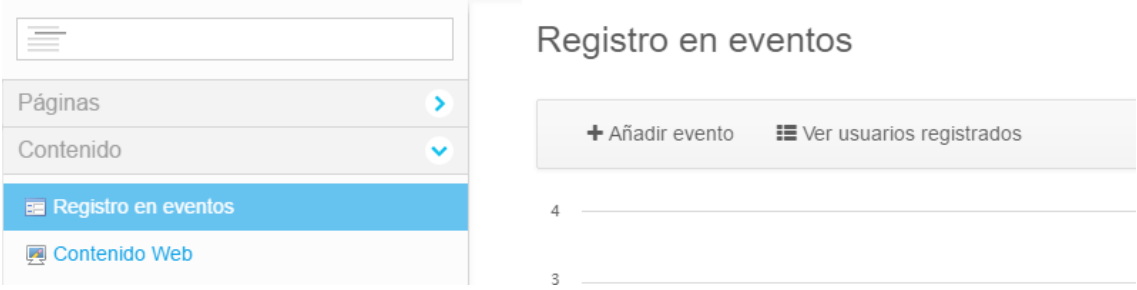
Para mejorar la aplicación resultante, se podrían implementar otras funcionalidades como son:

- Cancelar o editar registro: permitir a un usuario cancelar o modificar sus datos de registro.
- Versionado de eventos: habilitar un sistema de versionado de eventos de forma que se pueda mantener un control sobre los cambios que realiza cada gestor y recuperar versiones anteriores.
- Enviar por correo y compartir en redes sociales: permitir compartir la información del evento en redes sociales o por correo junto a un enlace al mismo.
- Implementar un sistema de pagos para los eventos que no son gratuitos.

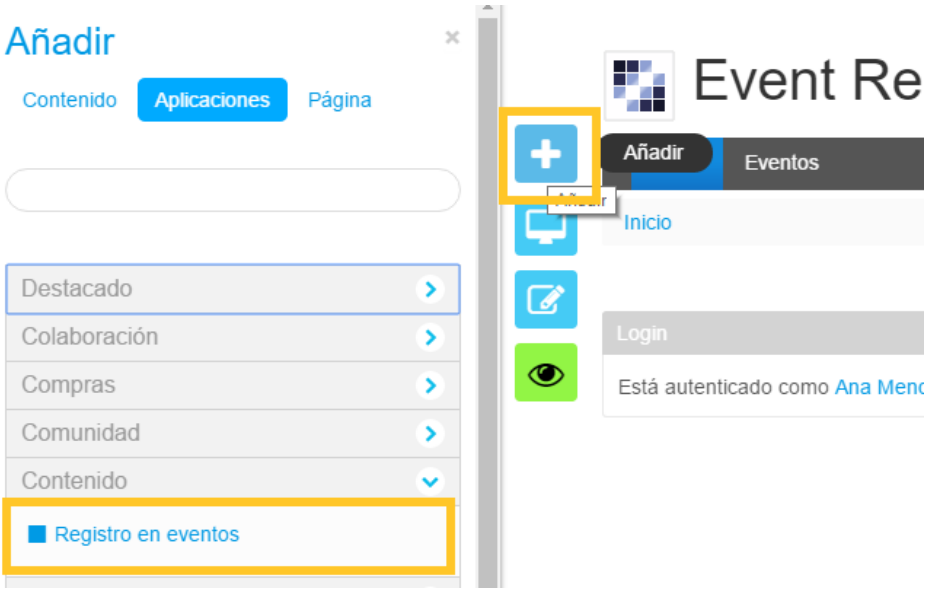
Documentación de despliegue

El despliegue de la aplicación en un servidor Tomcat se realiza de la forma habitual, copiando el archivo compilado .war a la carpeta *deploy* de la instalación de Tomcat con el bundle de Liferay 6.2 CE. El Service Builder de Liferay se encarga de crear la estructura de tablas necesarias según la descripción del modelo service.xml por lo que no es necesario ejecutar ningún script de base de datos. Los portlets también funcionarán con cualquier sistema de base de datos compatible con esta versión de Liferay Portal, incluso con un portal que use el sistema Hypersonic que hace uso de una base de datos en fichero y para el que no se requiere la instalación de un servidor de base de datos externo.

Una vez desplegado, se puede acceder al portlet de gestión de eventos desde la opción “Contenido” de cualquier site de nuestro portal:



Para poder utilizar el portlet público, es necesario que vayamos a una de las páginas de nuestro site y añadamos a la página este portlet desde la opción “Añadir” > “Aplicación” > “Contenido”:



Conclusiones del proyecto

Aunque la realización del proyecto no ha supuesto muchos problemas derivados de su dificultad ya que tengo experiencia previa en el desarrollo de portlets para Liferay, sí que ha supuesto un reto debido a la dedicación necesaria para su implementación; creo que las funcionalidades propuestas inicialmente resultaron demasiado ambiciosas y hubiera podido realizar una aplicación totalmente funcional dejando de lado algunas funcionalidades como las estadísticas de evento y las opciones de evento para añadir distintas sesiones. El haber planteado unas funcionalidades más realistas me hubiera permitido dedicar más tiempo a otras cuestiones de suma importancia en el desarrollo de proyectos de software como el desarrollo de pruebas unitarias, implementación de la configuración de los portlets, mejora del control de errores o poder establecer en la aplicación mayor control sobre los roles de usuario y las acciones que éstos pueden realizar.

El hecho de que haya escogido una aplicación basada en eventos también ha supuesto una dificultad mayor, ya que el tratamiento de las fechas en Java resulta algo engorroso.

La realización de este proyecto me ha servido para conocer mejor los componentes de presentación que ofrece la versión 6.2 de Liferay como los buscadores avanzados y también el uso de *Service Builder* con estructuras que contienen datos de tipo fecha, aunque el uso de componentes propios ha aumentado la curva de aprendizaje dada la escasa documentación sobre ellos. En algunos casos he tenido que revisar el código de los propios taglibs incluidos en el portal para saber cómo utilizarlos.

Como nota positiva, creo que gracias a la realización de este proyecto han mejorado mis habilidades en el desarrollo de portlets de Liferay ya que he aprendido cómo usar muchos componentes que no conocía, como la integración de un componente Search Container con una búsqueda avanzada.

Glosario

- **Portlet:** componente modular, similar a una aplicación que se despliega en un portal.
- **Liferay Marketplace:** catálogo de plugins para Liferay Portal accesible por web y a través del panel de control de Liferay. Lista distintos plugins desarrollados tanto por Liferay Inc como por desarrolladores o empresas no vinculadas a Liferay y permite instalarlos en nuestro portal.
- **Service Builder:** herramienta de generación de código mediante mapeo objeto-relacional (ORM) que genera las clases necesarias para la interacción de un portlet con una base de datos a partir de la definición de entidades en XML.

Referencias

- Liferay Developer Network: https://dev.liferay.com/
- Foros de Liferay: https://web.liferay.com/es/community/forums/-/message_boards/message-boards-home
- Khiêm Trần Blogs. Create Liferay portlet with localized input value: http://khiem-tran.blogspot.com.es/2014/04/create-liferay-portlet-with-localized.html
- AlloyUI v.2.0.x: http://alloyui.com/versions/2.0.x/
- Highcharts documentation: http://www.highcharts.com/docs
- Google Maps JavaScript API: https://developers.google.com/maps/documentation/javascript/?hl=es