JEE: Aplicación de gestión

Ignacio Baca Moreno-Torres
ITIS

Javier Ferró Garcia
19 jun. 2006

1 Índice

2 Resumen

El proyecto iniciado el pasado mes de febrero tenía como objetivos aprender el funcionamiento del desarrollo de una aplicación informática basada en JEE. El análisis, independiente de la tecnología usada en la implementación, propuso un esquema sencillo de una aplicación de gestión. Compuesto por una decena de casos de uso y un par de actores el proyecto cogía forma, aunque a la hora de describir el funcionamiento concreto de algunas de estas funciones hubo dificultades para separar la idea de una descripción no orientada a la programación, i.e. tendía ha describir más el "como" se iba a implementar, que el "que" realizar.

En la fase de diseño, he dedicado más de la mitad del tiempo a conocer JEE. Comenzando con J2EE en la plataforma Sun Application Server 8 y terminando en GlassFish v2 B7. Las primeras decisiones tras leer "J2EE Architect Handbook" y documentos relacionados con Struts e Hibernate era una aplicación basada en J2EE con persistencia gestionada por Hibernate y un capa MVC bajo Struts. Poco después comencé a leer la documentación de la última versión de JEE, este planteaba una solución estándar para estos problemas. Aportaba nuevas funcionalidades, pero de especial interés eran JPA y Faces y, más por comodidad, la inyección de recursos y el modelo simplificado de programación.

Una vez decidido todo respecto a que plataforma que iba a ser la base del desarrollo, realice el diseño de la aplicación. Una serie de entidades que daban la base persistente, unos EJB que contenían las reglas de negocio y unos componentes del interfaz de usuario de Faces formaban el diseño de la aplicación final. En este momento, y tras leer el tutorial de la ultima versión de java enterprise (documento que es prácticamente como leer el API pero narrado), parecía que el diseño basado en la plataforma escogida daría un resultado excelente.

Por ultimo la implementación. En el momento que el diseño tenía que convertirse en código hubo bastantes dificultades. La cantidad de información necesaria para controlar estos componentes (JPA,JSF,JTA…) era enorme, y dado que este modelo aporta mucha potencia (e.g. con una sencilla anotación como "Entity" creamos una entidad que se mapea de forma automática con una base de datos relacional), acompañado de un modelo sencillo hace un comienzo difícil. Como resultado, en esta fase he tenido cantidad de resultados inesperados y gran dificultad para analizar el flujo del estos (debug).

Al final he obtenido una aplicación funcional, algo pobre en lo referente a la lógica de negocio, pero amplia en el uso de recursos de JEE, aun así APIs importantes como JMS, JavaMail o todo lo referente a la seguridad se ha dejado fuera.

3 Cuerpo de la memoria

3.1 Introducción

3.1.1 Justificación y punto de partida:

El proyecto se iniciaba con unos conocimientos medios de JSE 1.4. Me parece que java es un leguaje tan sencillo como potente y conocer JEE era algo fundamental. Además el desarrollo de aplicaciones informáticas es conocido por su dificultad y especial mente a la hora de hacer estimaciones sobre esta, con el propósito, claro, de calcular un precio y una fecha. Plantear un proyecto, sus diferentes etapas y practicarlas eran importantes de cara a la experiencia como desarrollador.

En la parte de arquitectura del software tenía conocimientos obtenidos de asignaturas, pero ninguna experiencia profesional, así que mejorar el conocimiento en esta área era importante. Como base la documentación recomendada, planteaba ideas para el desarrollo en cada una de sus fases.

3.1.2 Objetivos

Los objetivos, ya descritos en el análisis, eran obtener conocimiento y experiencia del desarrollo del software. Para esto se propuso una aplicación de gestión, que serviría de objetivo preciso para alcanzar el difuso propósito del conocimiento de la tecnología JEE. A continuación enumero estos objetivos:

- ✓ Aprender el funcionamiento de la plataforma JEE y sus tecnologías: WEB services, EJB, XML, Servlets…
- ✓ Utilizar esta plataforma para el desarrollo de una aplicación informática de administración y comunicación de un club social.
- ✓ Adquirir una visión del trabajo del arquitecto técnico, analista de negocios, creador de modelos de datos y otros responsables de proyecto.
- ✓ Conocer el ciclo de vida del desarrollo de una aplicación informática, en concreto una aproximación del ciclo RUP o XP.
- ✓ Conocer el funcionamiento de una herramienta CASE como NetBeans, y las herramientas de que dispone como diagramas UML, documentación y gestión de calidad (en el aspecto del rendimiento) para aplicaciones en J2EE.

3.1.3 Alcance del proyecto

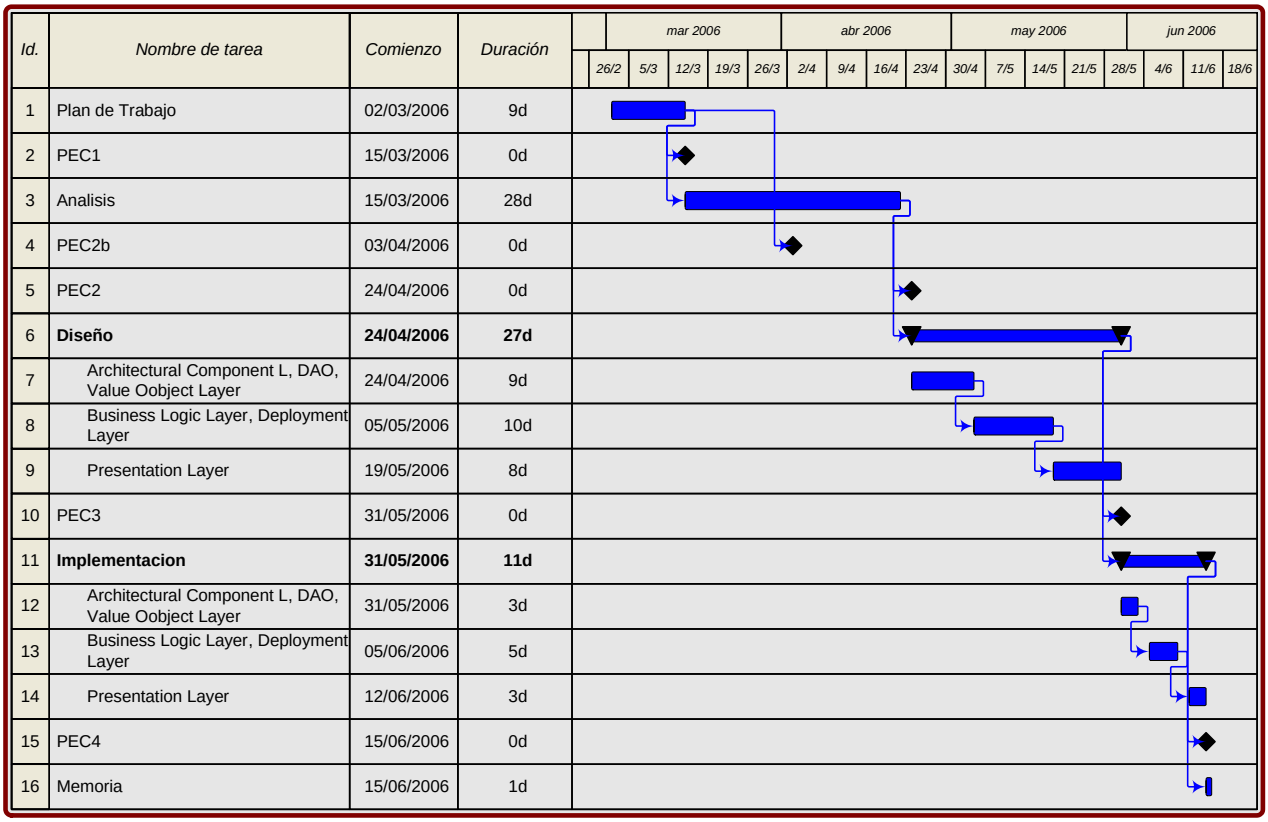
El método que se ha seguido para conseguir estos objetivos que se han propuesto ha sido el desarrollo de una aplicación, que además es parte de los objetivos. Para esto, se definieron unos "sub-objetivos" a los que la aplicación debía dirigirse, esto se considero el alcance del proyecto y sirvió como guía para poder desarrollar los diferentes elementos y así conocer como JEE soluciona o facilita estas labores.

Construir un sistema que mejore la comunicación de un club social y ayude a la administración y gestión de las actividades. El proyecto debe permitir a los socios hacer lo siguiente:

- ✓ Gestionar los socios y permitir su comunicación.
- ✓ Gestionar actividades y los recursos que necesitan.
- ✓ Registrar la asistencia de los socios a las actividades y generar informes económicos.
- ✓ Crear noticias y foros de comunicación.
- ✓ Auto-crear noticias de actividades.
- ✓ Crear informes de estado del club.
- ✓ Creación y publicación de documentos.

3.1.4 Planificación

La distribución temporal del proyecto esta basada en las fechas de entrega de las PEC. Así el proyecto comienza el día 02/03/2006 y termina el



Id.	Nombre de tarea	Comienzo	Duración
1	Plan de Trabajo	02/03/2006	9d
2	PEC1	15/03/2006	0d
3	Analisis	15/03/2006	28d
4	PEC2b	03/04/2006	0d
5	PEC2	24/04/2006	0d
6	**Diseño**	**24/04/2006**	**27d**
7	Architectural Component L, DAO, Value Oobject Layer	24/04/2006	9d
8	Business Logic Layer, Deployment Layer	05/05/2006	10d
9	Presentation Layer	19/05/2006	8d
10	PEC3	31/05/2006	0d
11	**Implementacion**	**31/05/2006**	**11d**
12	Architectural Component L, DAO, Value Oobject Layer	31/05/2006	3d
13	Business Logic Layer, Deployment Layer	05/06/2006	5d
14	Presentation Layer	12/06/2006	3d
15	PEC4	15/06/2006	0d
16	Memoria	15/06/2006	1d

16/06/2006 con esta entrega, la memoria. Para las etapas de diseño e implementación se ha dividido en varías tareas correspondientes a las capas de las que se compondrá la aplicación.

Los hitos coinciden con las PEC, exceptuando los hitos PEC2b que corresponde con la revisión del análisis y la PEC4 que indica la finalización de la tarea de implementación. Como el equipo de desarrollo está formado por una única persona he utilizado el método en cascada y no se ha puesto ninguna

tarea paralela a otra. En consecuencia, el camino crítico coincide con la totalidad del proyecto.

3.1.5 Productos obtenidos

El resultado final, es una página Web desde la que se puede, en cierto modo, gestionar un club social. Esta "Web" esta compuesta por dos módulos java EE. Uno para presentar la información y debe almacenarse en un contenedor Web. Este modulo es el que da forma al producto final, ya que es lo que un cliente "ve". El otro modulo es el encargado de la lógica de negocio y debe desplegarse en un contenedor EJB.

Dentro de esto módulos se pueden destacar los siguientes elementos.

- ✓ Los diferentes archivos JSP que sirven para configurar el interfaz de usuario.
- ✓ Los archivos XML de despliegue, que dan información de configuración para agregar estos módulos a contenedores.
- ✓ Los java beans del modulo Web, que permiten una comunicación sencilla desde los JSP hacia las clases.
- ✓ Las entidades del modulo EJB, que relacionan las clases con tablas de bases de datos.
- ✓ Los SessionBeans del modulo EJB, que tratan la lógica del negocio y la comunicación remota.

3.1.6 Descripción de los capítulos

A continuación se muestran los capítulos Análisis y Diseño. Presenta la información obtenida en los pasos de mismo nombre.

El análisis, primera fase de un proyecto informático, presenta información del objetivo que se quiere alcanzar con el software. En nuestro caso aporta datos sobre el funcionamiento de un club social, así como los roles que sus miembros tienen y las acciones que pueden o no pueden realizar. En esta fase se hace también una planificación temporal, y dado que el PFC se basa en este proyecto de software para alcanzar sus objetivos, la planificación obtenida en el análisis se muestra como la planificación del proyecto. De forma similar el alcance del proyecto se muestra fuera del análisis.

En el segundo capitulo que veremos a continuación muestra el diseño. Este intenta explica de forma inequívoca las acciones descritas por el análisis. Para lograrlo se apoya en UML que permite explicar y construir de forma grafica modelos del sistema, esquemas de bases de datos, procesos de negocio y otros aspectos. Principalmente usaremos los diagramas de clases que define este lenguaje.

3.1.7 Desarrollo de software

Para entender como funciona JEE y que puede hacer por nosotros se planteó desarrollar un software. Para desarrollar software hay varios métodos, la mayoría basados en el clásico método de cascada, en concreto se escogió el RUP

que es, hoy en día, el método estándar junto con UML para desarrollar aplicaciones.

A continuación se muestran los resultados de dos fases importantes en el desarrollo de software, el análisis y el diseño.

3.2 Análisis

3.2.1 Casos de uso

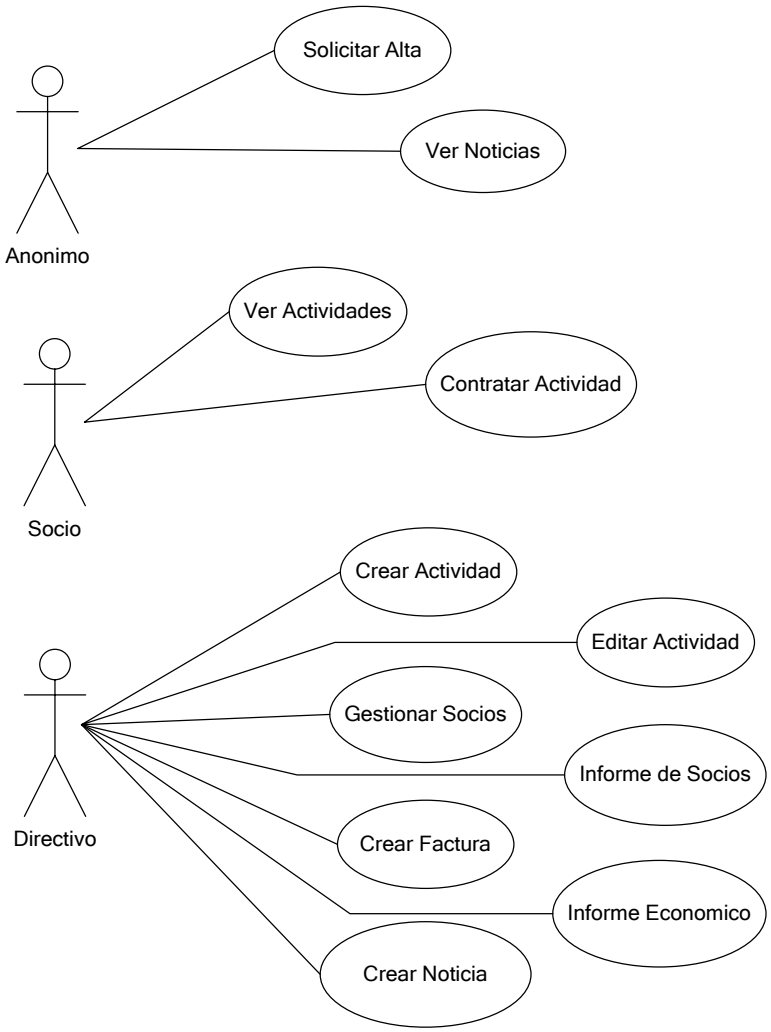
A lo largo de los aparatos siguientes se irán identificando los casos de uso.

3.2.1.1 Actores

Los actores son tres: anónimo, socio y directivo. El actor "anónimo" está relacionado con los clientes esporádicos que quieren obtener información del club de forma rápida y sencilla. Este actor no necesita ser identificado. El actor "socio" representa a los miembros del club. Este actor debe estar registrado y autentificado. Las acciones que puede realizar están descritas en los casos de uso. El "directivo" equivale a la junta directiva y es una extensión de las posibilidades de un "socio".

3.2.1.2 Diagrama de casos de uso

A continuación se muestra el diagrama de casos de uso de la aplicación:


Solicitar Alta
Ver Noticias
Anonimo
Ver Actividades
Contratar Actividad
Socio
Crear Actividad
Editar Actividad
Gestionar Socios
Informe de Socios
Crear Factura
Informe Economico
Crear Noticia
Directivo

3.2.1.3 Documentación Textual

CU-01 : Solicitar Alta

Resumen de funcionalidad: registra un socio en la base de datos

Actores: anónimo

Precondición: El socio no debe estar dado de alta.

Poscondición: El socio queda almacenado pero pendiente de confirmación.

Descripción: El socio anónimo debe introducir sus datos (nombre, apellidos, fecha de nacimiento, DNI, Telf. y e-mail). En caso de ser menor de edad deberá introducir los datos del padre (mismos campos).

Alternativas y excepciones:

CU-02 : Ver Noticias

Resumen de funcionalidad: Da acceso a las noticias almacenadas en la base de datos.

Actores: anónimo

Precondición: no hay precondiciones.

Poscondición: ninguna.

Descripción: permite ver un listado de noticias mostrando únicamente una introducción. Este listado esta ordenado por fecha. También permite la visualización completa de una única noticia.

Alternativas y excepciones:

CU-03 : Ver Actividades

Resumen de funcionalidad: muestra una lista de las actividades.

Actores: socio.

Precondición: ninguna.

Poscondición: ninguna.

Descripción: permite ver la lista de actividades mostrando un resumen de la información. El socio puede elegir la forma de ordenar y podrá aplicar filtros del estado de la actividad (realizada, pendiente, cancelada). También debe mostrar la información completa.

Alternativas y excepciones: debe darse la opción de contratar una actividad de la lista según el caso de uso Contratar actividad.

CU-04 : Contratar actividad

Resumen de funcionalidad: registra una nueva actividad en la base de datos.

Actores: directivo.

Precondición: la actividad no debe estar en la base de datos.

Poscondición: la actividad debe quedar almacenada en la base de datos y debe haberse creado una noticia de esta actividad.

Descripción: solicita la información de la actividad (nombre, lugar, fecha, precio, plazas, comentarios). Una vez introducida la información se creará una noticia con la información de la actividad según el caso de uso Crear Noticia.

Alternativas y excepciones:

CU-05 : Editar actividad

Resumen de funcionalidad: permite modificar una actividad ya registrada.

Actores: directivo.

Precondición: la actividad debe existir en la base de datos.

Poscondición: la nueva información queda guardada en la base de datos. Si se modifica el precio o se cancela la actividad debe reflejarse en las facturas. Las noticias quedan también actualizadas.

Descripción: permite la modificación de los campos de una actividad. En caso de modificar el precio de una actividad que ya tenga socios apuntados deberán crearse facturas que reflejen este cambio. En caso de que el precio sea menor se crearán devoluciones.

CU-06 : Gestión de socios

Resumen de funcionalidad: permite aceptar y denegar a los socios.

Actores: directivo.

Precondición: el socio debe estar registrado en la base de datos.

Poscondición: el socio queda modificado en la base de datos.

Descripción: la principal función es aceptar las solicitudes de los socios tras registrarse según el caso de uso Solicitar alta. También permite denegar el acceso a un socio.

CU-07 : Informe de socios

Resumen de funcionalidad: muestra una lista de socios.

Actores: directivo.

Precondición: ninguna.

Poscondición: ninguna.

Descripción: permite ver una lista de los socios del club. También muestra datos estadísticos como el nº de socios, nº de socios por categoría y la media de edad de los socios.

CU-08 : Crear Factura

Resumen de funcionalidad: crea una factura.

Actores: directivo.

Precondición: la factura no está registrada en la base de datos.

Poscondición: la factura queda registrada en la base de datos.

Descripción: permite introducir los datos de un factura (numero único anual, año, fecha de creación, cantidad, facturada por, en concepto de, categoría). En caso de que la factura esté relacionada un socio deberá registrarse el identificador del socio. Si es una factura para una excursión deberá registrarse también el identificador de la excursión. La aplicación podrá crear facturas que no se hallan procesado (i.e. que no se halla realizado ningún pago entre los afectados) y por tanto debe darse la posibilidad de cambiar el estado con posterioridad.

Alternativas y excepciones:

CU-09 : Informe económico

Resumen de funcionalidad: muestra una lista de facturas y un balance.

Actores: directivo.

Precondición: ninguna.

Poscondición: ninguna.

Descripción: permite crear un informe económico mostrando tanto un resumen de facturas como un balance total. Permite elegir la fecha de inicio y fin del informe. Se distinguirán los ingresos de los gastos diferenciándolos por colores. También se mostrará un resumen de gastos según categorías. Se podrá elegir no mostrar el resumen de facturas.

CU-10 : Crear Noticia

Resumen de funcionalidad: añade una noticia nueva.

Actores: directivo.

Precondición: la noticia no debe existir en la base de datos.

Poscondición: se añade la noticia a la base de datos.

Descripción: crea una noticia y la almacena en la base de datos. Los campos de la noticia son titulo, contenido, publicado por, fecha de caducidad y

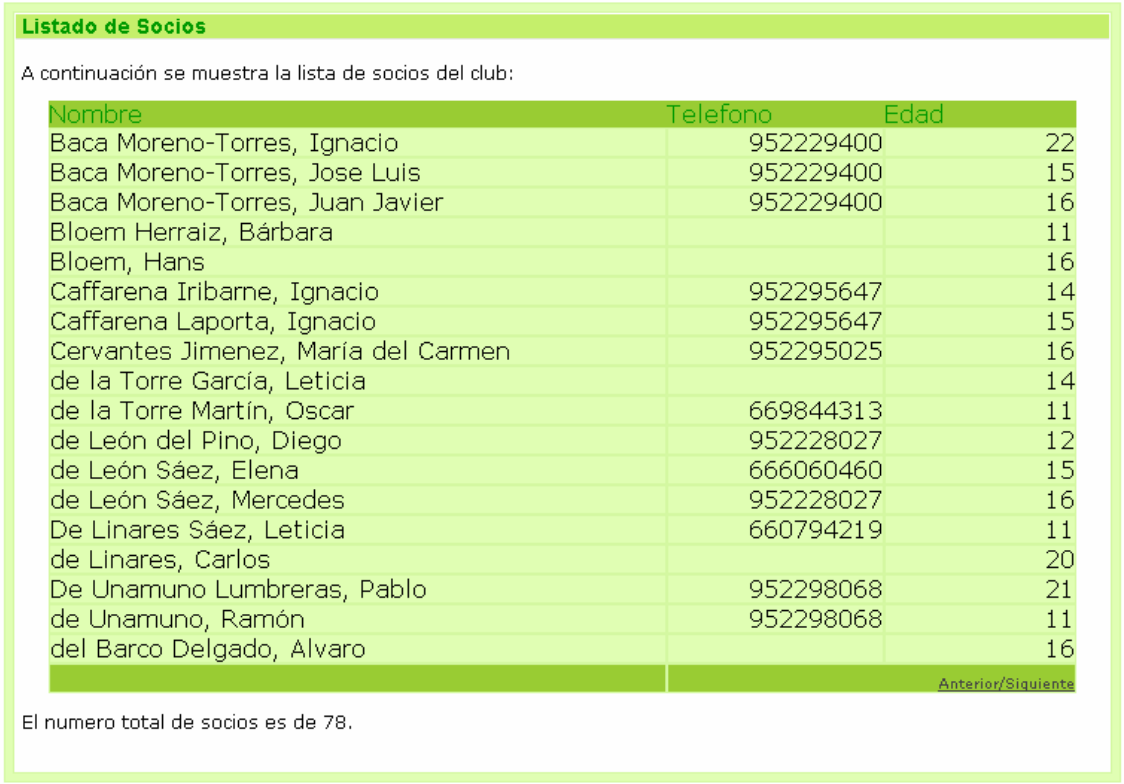
categoría. En caso de estar relacionada con una actividad deberá registrarse la actividad.

3.2.2 Análisis del interfaz

El interfaz de la aplicación será por medio de HTML. Estará basada en la actual página del club. A continuación se pueden ver unos prototipos que muestran como será de forma aproximada el resultado final de la aplicación basad en Web.

En la zona de menú se mostraran las diferentes zonas de actuación como son la de actividades, socios o facturas. Los menús se mostrarán dependiendo del usuario que esté manejando la aplicación.

A la hora de editar y mostrar la información aparecerá en la zona de "Cuadro principal". A continuación muestro el contenido de la "Zona de actuación para Listado de Socios.



Listado de Socios

A continuación se muestra la lista de socios del club:

Nombre	Telefono	Edad
Baca Moreno-Torres, Ignacio	952229400	22
Baca Moreno-Torres, Jose Luis	952229400	15
Baca Moreno-Torres, Juan Javier	952229400	16
Bloem Herraiz, Bárbara		11
Bloem, Hans		16
Caffarena Iribarne, Ignacio	952295647	14
Caffarena Laporta, Ignacio	952295647	15
Cervantes Jimenez, María del Carmen	952295025	16
de la Torre García, Leticia		14
de la Torre Martín, Oscar	669844313	11
de León del Pino, Diego	952228027	12
de León Sáez, Elena	666060460	15
de León Sáez, Mercedes	952228027	16
De Linares Sáez, Leticia	660794219	11
de Linares, Carlos		20
De Unamuno Lumbreras, Pablo	952298068	21
de Unamuno, Ramón	952298068	11
del Barco Delgado, Alvaro		16

Anterior/Siguiente

El numero total de socios es de 78.

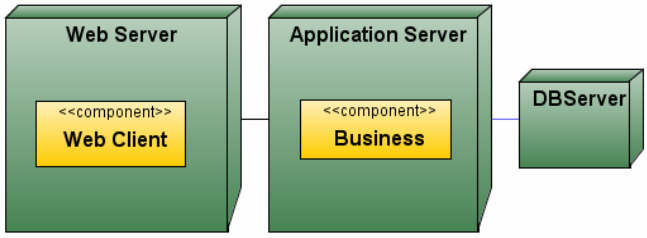
La estructura propuesta es sencilla, basada principalmente en estilos CSS con poca carga de imágenes. Se intentará dar un aspecto agradable y sencillo siguiendo la filosofía de la organización. En caso de problemas o notificaciones se mostraran de forma discreta al final de la página.

3.3 Diseño

Este apartado muestra la estructura final del proyecto a través de diagramas UML que nos permiten ver las relaciones entre los diferentes elementos que forman el sistema.

3.3.1 Una visión global

El sistema está dividido en dos partes principales que corresponden con el esquema básico de una aplicación basada en JEE (i.e. un servidor de paginas Web que recibe las solicitudes de un cliente y las procesa a través de servlets o



JSPs y un servidor de aplicación que administra un contenedor de EJBs donde se encuentra la lógica de negocio).

El componente "Web Client" contiene todo lo necesario para que un cliente pueda acceder y manipular la información del sistema. Para esto el componente está basado en el framework JSF que permite la creación y gestión de clientes del lado del servidor. Para construir el interfaz de usuario basado en componentes se utilizaran los JSP, otra tecnología java entreprise que permite generar servlets o en nuestro caso combinar componentes faces a través de archivos XML.

En el componente "Business" contiene toda la lógica de negocio, la administración de persistencia de las diferentes entidades y la seguridad. Esta, debido a los requerimientos de un sistema self-register (i.e. permite a los usuarios crearse cuentas nuevas) y dado que los sistemas de control de "roles" y "principales" son dependientes del servidor de aplicación utilizado, se ha creado uno sencillo que nos permite alcanzar los objetivos del proyecto y no depender de una implementación especifica de JEE.

3.3.2 Modelo de objetos

Para el modelo de objetos se ha utilizado una división en capas basada en el libro J2EE: Architect's handbook que contiene seis capas; presentación, despliegue, negocio, acceso a datos, objetos detalle y componentes de diseño. Aunque la aplicación únicamente utiliza cinco, para esto fusiona las capas de

negocio y despliegue en una única capa. Las capas a su vez coinciden con los componentes, siendo el componente "Web Client" la capa de presentación y el componente "Business" las capas de negocio y acceso a datos. La capa de objetos detalle ("value objects") debe estar compartida en el sistema completo y por tanto suele ser un componente separado (i.e. un archivo .jar que se agrega como librería), pero para el desarrollo se ha implementado como paquete del componente "Business". La capa de diseño no contiene ningún desarrollo para esta aplicación en concreto y contiene únicamente las diferentes herramientas que proporciona el API de JSE y JEE, así que no se describirá en el diseño.

La distribución final de capas queda de la siguiente forma:

- ✓ Capa de presentación o "Presentation Tier"
- ✓ Capa de negocio o "Bussines Tier"
- ✓ Capa de acceso a datos o "Integration Tier"

A continuación se muestra algunos diagramas UML que permiten ver cada una de las capas con más detalle comentando los casos más relevantes. Para poder ver la descripción completa de cada una de las clases que compone cada capa se deberá acceder a través del informe HTML de cada uno de los esquemas.

3.3.2.1 Capa de presentación

Esta capa da acceso a la aplicación a partir de un cliente Web. Este accede a través del servidor que atiende las llamadas utilizando la tecnología de Servlets. Para atender estas llamadas de los clientes, y con el propósito de implantar un patrón MVC en la capa de presentación el proyecto utiliza el Framework JSF (Java Server Faces). Está tecnología utiliza un sistema parecido al de AWT, tratando los diferentes elementos visuales como componentes, orientado para servicios Web (pero no exclusivo para estos) aporta unos componentes que representan objetos HTML y que permiten el desarrollo de un interfaz de componentes representado en HTML. Además para desarrollar este interfaz permite usar la tecnología JSP, construyendo de forma sencilla la interfaz a partir de tags que representan estos componentes, y que luego serán representados por código HTML. JSF también permite crear validadotes, convertidores, acciones y listeners que hacen posible la comunicación entre la capa "View" y la capa "Model". El "controler", elemento de unión entre las otras capas, también lo aporta JSF, y en este proyecto se utiliza sin modificaciones el `javax.faces.webapp.FacesServlet`, que hace de "Front Controler". Para configurarlo se utiliza el archivo de configuración de Faces.

En la capa modelo se utiliza un bean administrado por faces, que permite su inicialización y el control de contexto. Este "Managed Bean" es `UserSession`, y como su nombre indica esta en el contexto de sesión.

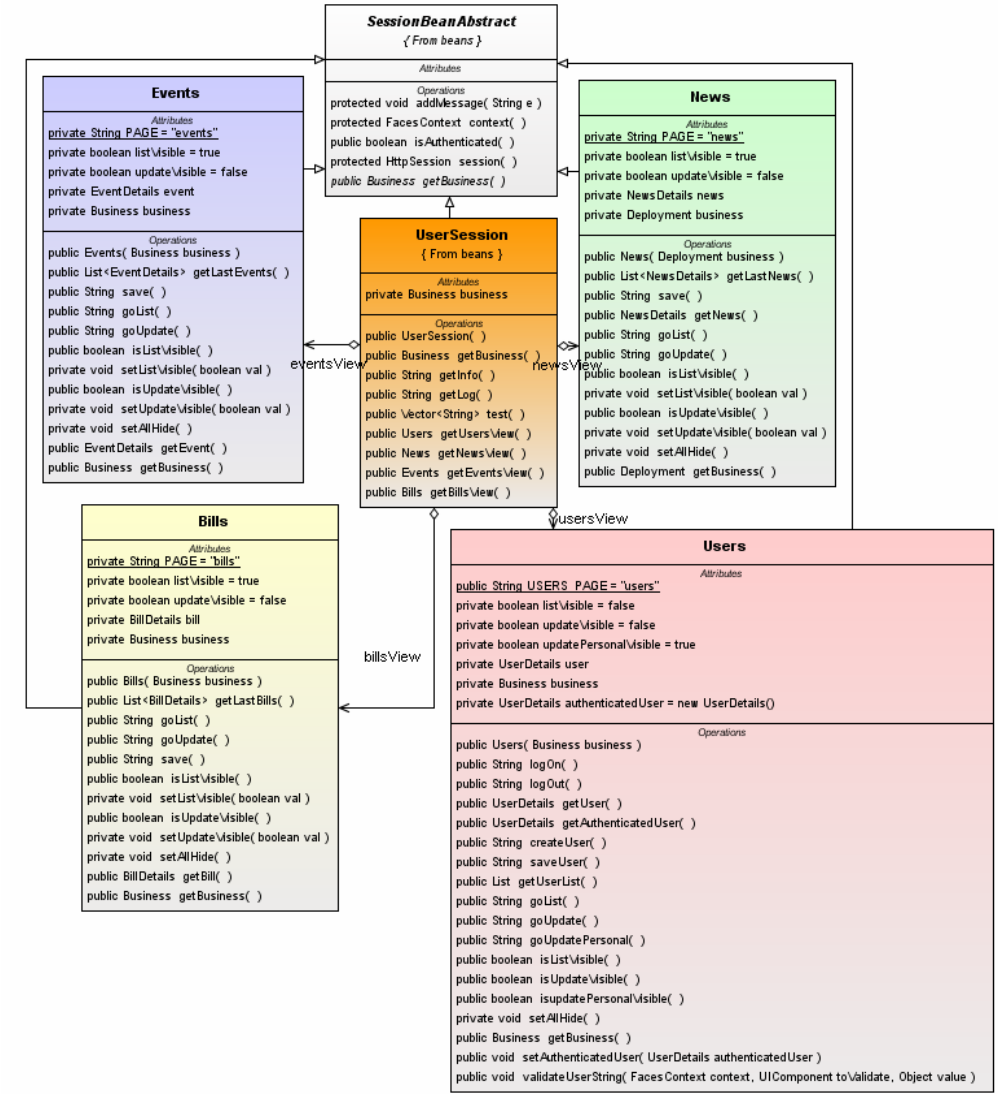
A continuación se muestra un esquema de las clases implementadas en esta capa, la clase `SessionBeanAbstract` es una clase de utilidad, y extienden de ellas el resto. `UserSession` es el `managedBean` con el que se comunican los

componentes escritos en JSP y las cuatro clases `Events`, `News`, `Bills` y `Users` sirven para organizar los datos y métodos y están relacionados con paginas JSP, así por ejemplo la pagina de facturas utiliza los métodos y propiedades contenidos en `Bills`. El propósito es un acceso sencillo a las diferentes opciones que permite el modelo, así el diseñador podrá mostrar la lista de noticias utilizando el EL de java de la siguiente forma;

```
action="#{userSession.newsView.goList}"
```

O usar el validador de cadenas de nombre de usuario usando;

```
validator="#{userSession.usersView.validateUserString}".
```

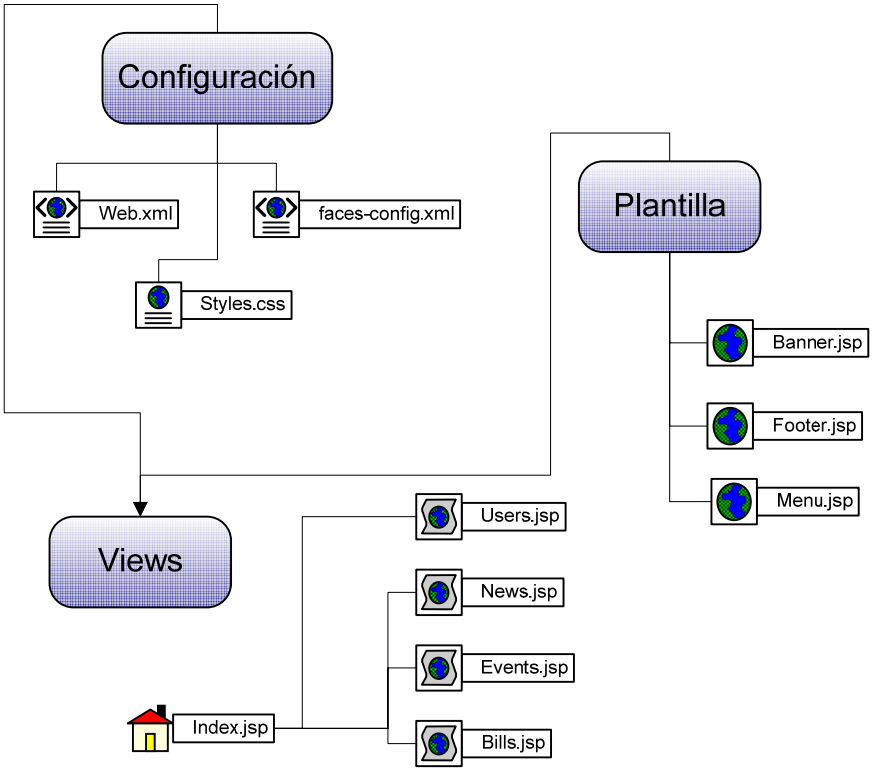


Esta capa esta formada en gran parte por otros archivos que estructuran y configuran las capas VIew y Controler. La capa View esta integrada principalmente por archivos JSP, que contienen los componentes de Faces. Este, al igual que AWT, esta estructurado siguiendo un patrón basado en Composite y estas composiciones parten del elemento View. Como se ve en el esquema que se muestra a continuación, cada una de las paginas JSP tiene un View a partir del cual se compone el espacio de trabajo. Para omitir los componentes

comunes se ha utilizado el TAG include de JSP para llamar a los diferentes elementos de plantilla.

También son importantes los archivos de configuración, tanto el archivo de despliegue, en el que configuramos un filtro para que las solicitudes de los clientes se dirijan al FrontControler de Faces como el archivo de configuración de Faces, en el que se configuran las diferentes reglas de navegación así como los Managed Beans.

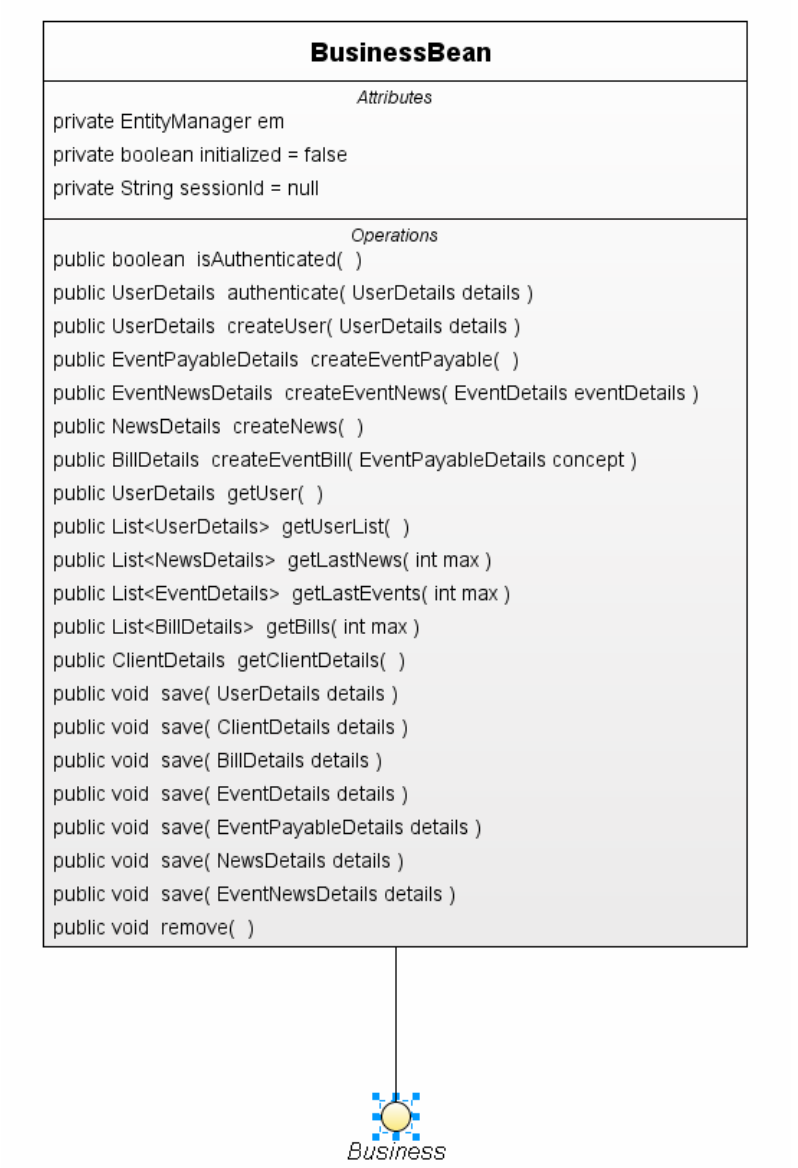
El esquema que representa estos elementos de la capa de presentación queda de la siguiente forma.



3.3.2.2 Capa de negocio

La capa de negocio utiliza una única clase que representa un session de usuario a través de un EJB Session Statefull Bean. Esta clase de interfaz remoto permite realizar todas las funciones de la aplicación.

Como su interfaz es remota, para mejorar el rendimiento en las comunicaciones se utiliza la capa VO, de esta forma los valores devueltos y recibidos por la interfaz están encapsulados en estos objetos que por conveniencia se llaman igual que las entidades seguidos de un sufijo “details”.



3.3.2.3 Capa de acceso a datos - Entidades

Para la persistencia se utiliza el API de persistencia que trae JEE 1.5. Cada una de las clases que se muestran a continuación implementan el interfaz `@Entity`, excepto `Payable`.

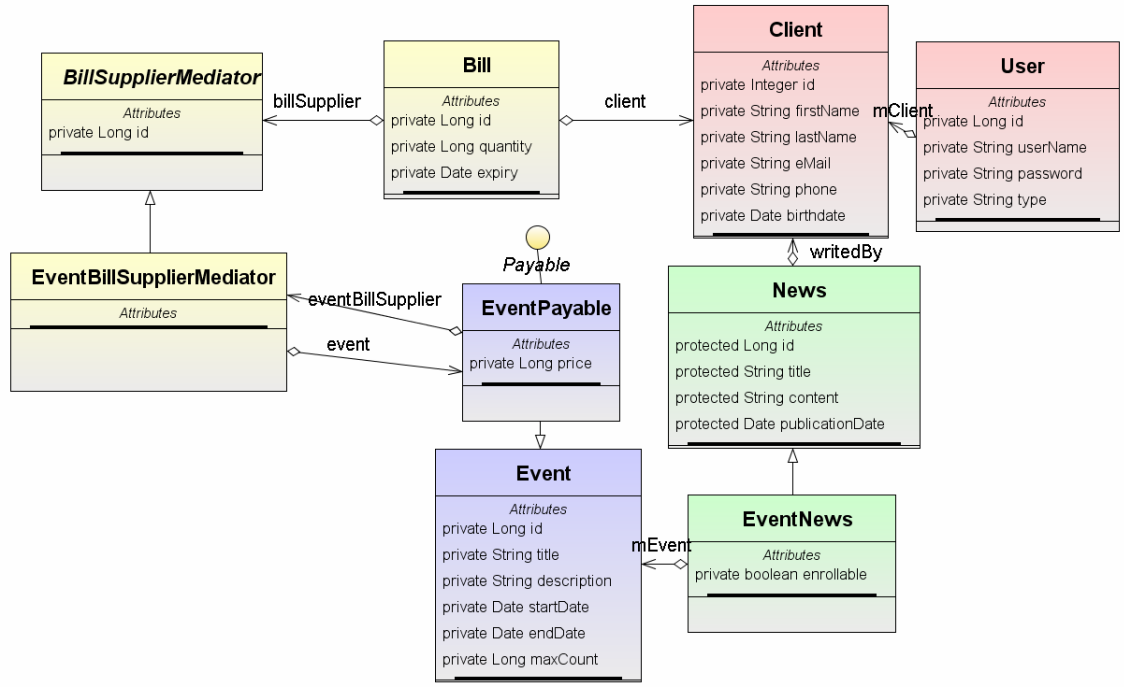
La clase `User` se utiliza como base de sistema de autentificación a partir de un nombre de usuario y contraseña, una vez autentificado se obtiene el cliente que contiene la información detallada de cada usuario. El usuario también contiene el tipo de cliente que podrá ser anónimo, socio o directivo (i.e. cada uno de los actores).

Para las facturas se ha utilizado una clase `Bill`. Para representar objetos que pueden ser pagado se ha utilizado el interfaz `Playable`, como una entidad no puede ser representada por una interfaz y dado que el propósito es que el interfaz permita a `Bill` estar relacionado con cualquier tipo de objeto pagable se han utilizado las clases mediator que permiten que cada factura este asociado

con un pagable al que accede a través de la clase abstracta `BillSupplierMediator`, y cada pagable implementa su propio mediator.

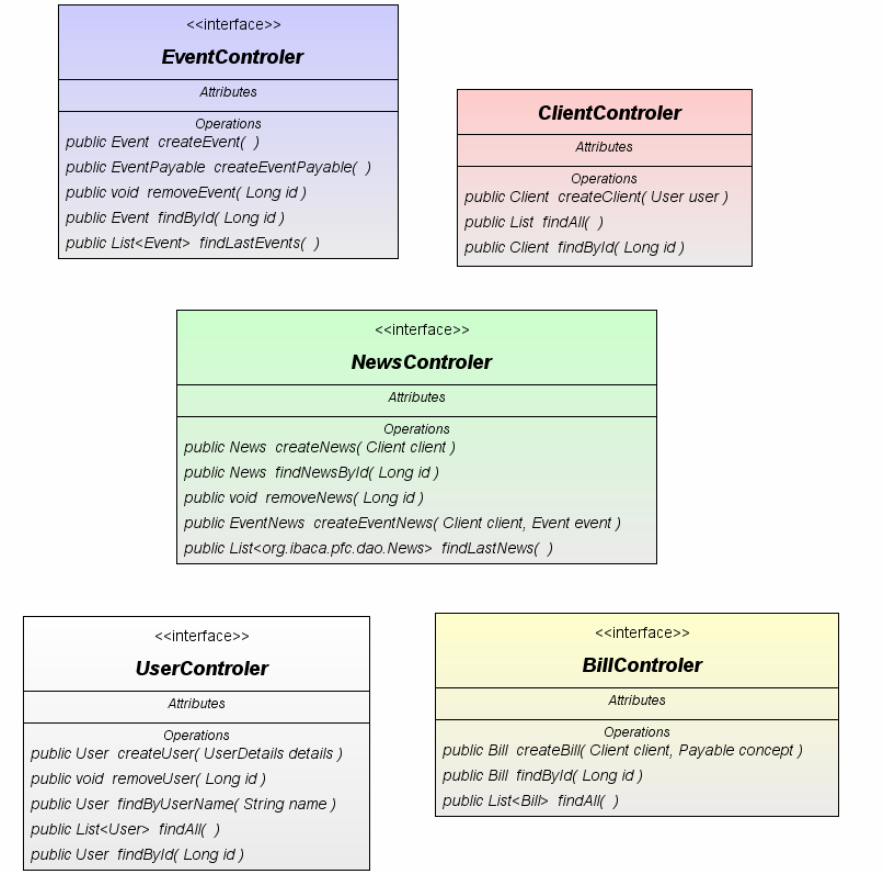
Las noticias además de poder ser agregadas por los usuarios del sistema, deben de poder auto generarse al crear una excursión o evento, con este fin se ha creado una clase hija de `News` que permite estar relacionada con el evento que anuncia de forma que en la capa de negocio se puede controlar y actualizar cambios entre la noticia y el evento.

A continuación se muestra el diagrama de clases para el paquete `Entity`.



3.3.2.4 Capa de acceso a datos - Fachada

Para ocultar del resto de capas las operaciones de acceso a los datos se han implementado los siguientes `session beans` que hacen de fachada (`Facade`) de las entidades de la capa de acceso de datos. Cada uno de los SB actúa sobre su o sus entidades y su función es controlar las operaciones CRUD de cada uno de estos, esto en algunos casos abarca mas de una entidad, como es el caso de la creación de un usuario, que llama al controlador de cliente y crea un cliente de forma transparente para poder mantener el estado correcto de las entidades (ya que un usuario siempre esta relacionado con un cliente).

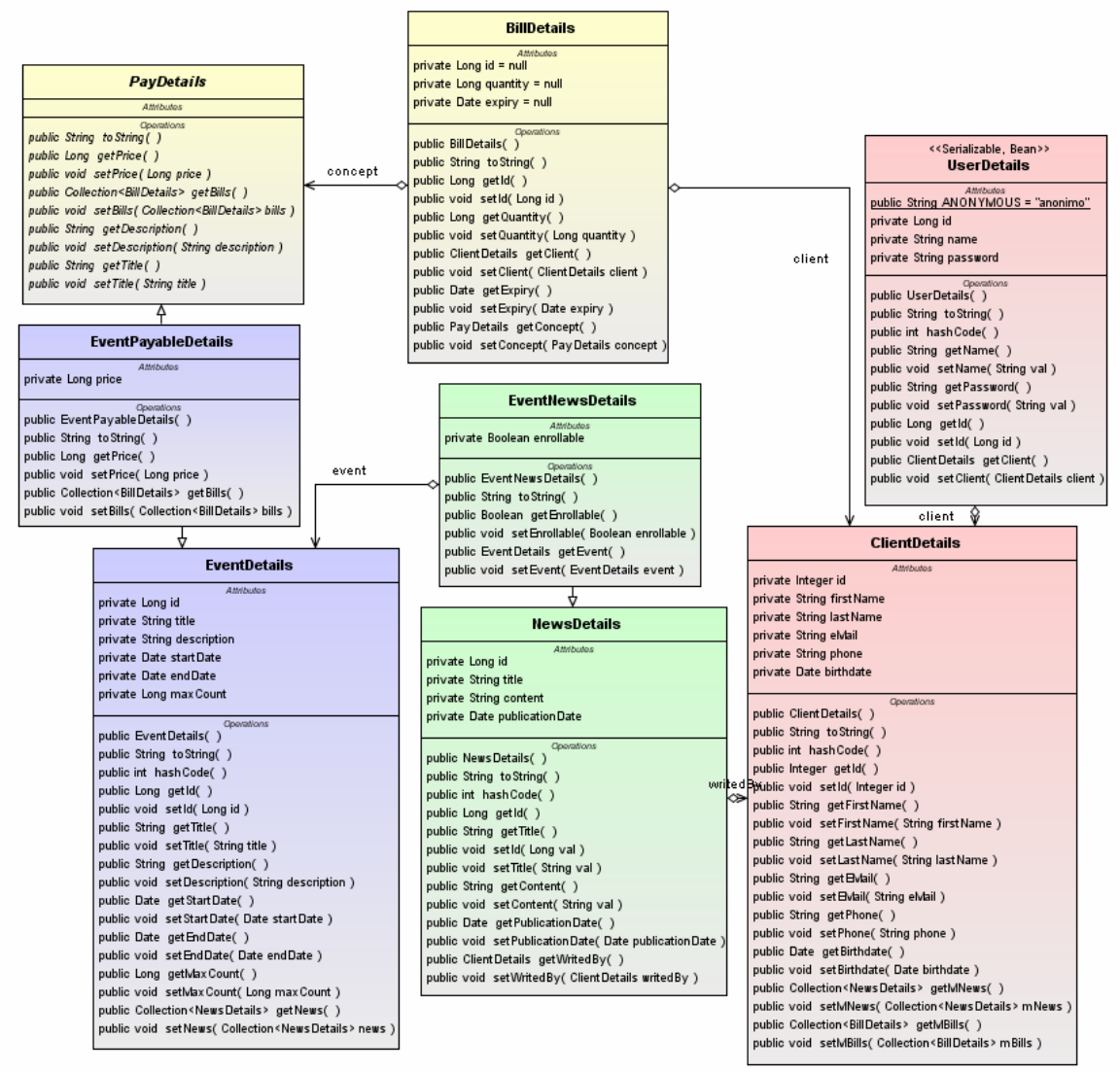


3.3.2.5 Capa de objetos detalle

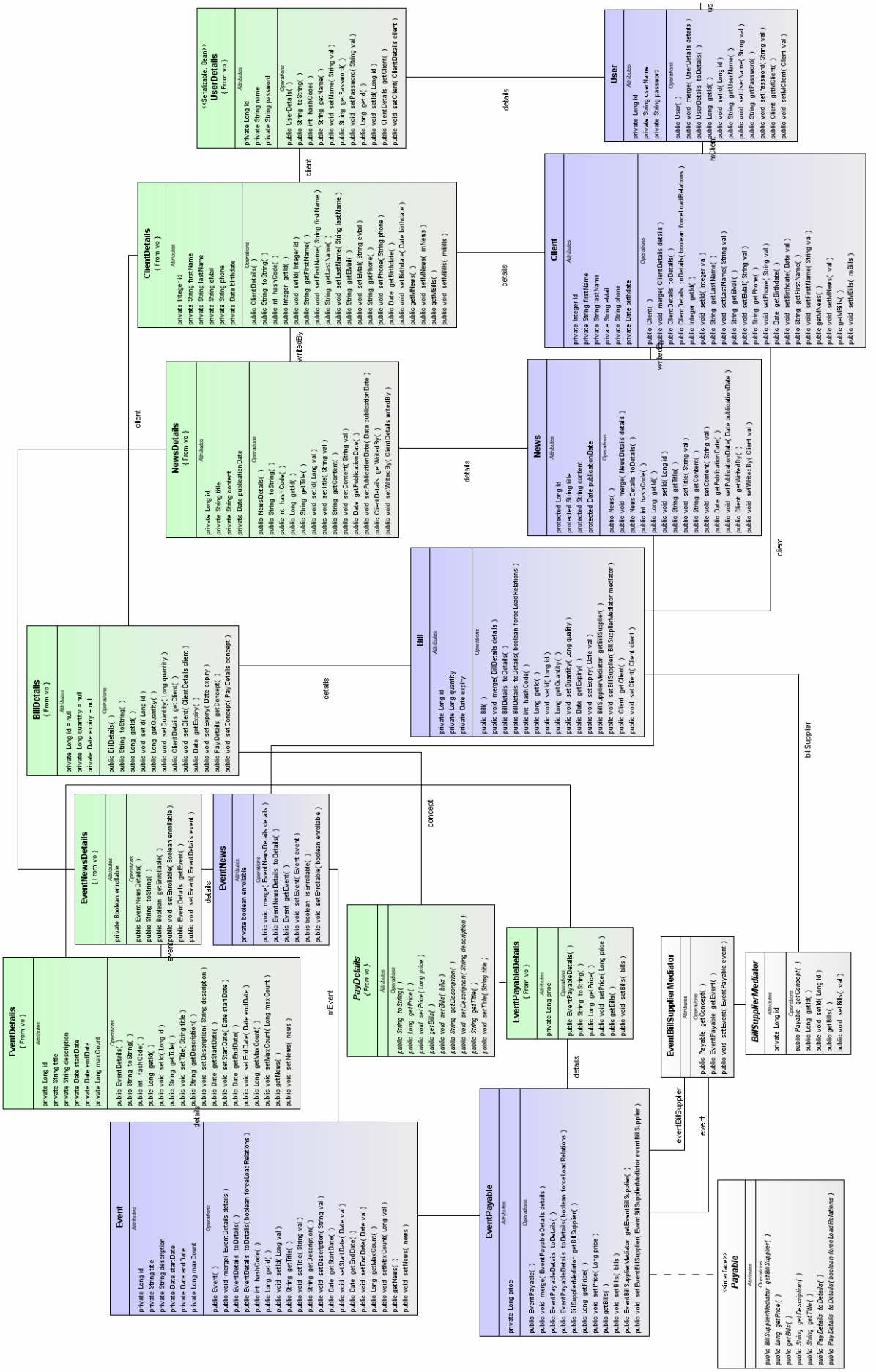
Esta capa, como ya se ha comentado, se usa para la comunicación remota. Representa cada una de las entidades y permite crear relaciones de la misma forma que las entidades están relacionadas. Para crear estos objetos a partir de una entidad se ha implementado en cada entidad el método toDetails, así como el método merge, para combinar los valores no nulos ni relacionados (i.e. valores detalles de otras entidades contenidas en la entidad que se pretende combinar) que contiene el objeto detalle en la entidad correspondiente.

Todos los objetos de la capa detalle siguen el estándar JavaBeans que permite el acceso a propiedades a través de métodos get<property> o set<property>. Esto es especialmente útil en la capa presentación, ya que podremos acceder a los diferentes valores de forma sencilla a través del lenguaje de expresiones unificado de Java.

Excepto la clase Payable que es abstracta, el resto son clases y todas implementan el interfaz serializable.



Por ultimo se muestra un esquema en el que se relacionan, como se comento anteriormente, las entidades con los objetos de detalles que se utilizan para la comunicación remota. Observar los métodos merge y toDetails que se encuentran en cada entidad y proponen una solución para la correcta trasformación entre un tipo y otro.


«Serializable, Bean»
UserDetails
{ From vo }
User
ClientDetails
{ From vo }
Client
NewsDetails
{ From vo }
News
BillDetails
{ From vo }
Bill
EventNewsDetails
{ From vo }
EventNews
EventDetails
{ From vo }
Event
PayDetails
{ From vo }
EventPayableDetails
{ From vo }
EventPayable
EventBillSupplierMediator
BillSupplierMediator
«interface»
Payable
client
details
writtedBy
event
concept
billSupplier
eventBillSupplier

3.4 El componente EAR

En la última etapa del proyecto se pretendía pasar del diseño a la implementación, y es esta implementación la que da como resultado el archivo de extensión EAR. Está etapa ha sido bastante complicada y empezó antes de terminar el diseño. El componente EAR, explicado de forma mas precisa, es la forma que java ha establecido para encapsular la información necesaria para el despliegue de un aplicación JEE en un servidor de aplicaciones. Un componente EAR a su vez puede tener otros componentes de tipo EJB, WAR, o librarías de java de formato JAR.

La implementación está dividida, como se muestra en el diseño, en dos componentes, uno de cada tipo (WAR y EJB). Como herramienta de desarrollo se utilizo NetBeans 5.5beta que da soporte a la ultima versión de JEE 5, además de permitir ingeniería inversa en diagramas UML y otras características que hacían de este software una herramienta ponente y eficiente para nuestro objetivo. Para la persistencia, o capa DAO en el diseño, se uso JPA que en la versión de java application Server 9 daba problemas en la herencia de objetos abstractos al no encontrar lo atributos heredados de forma correcta. Esto provoco el uso de la última versión de comunidad de este servidor llamada glassfish. Concretamente la implementación y pruebas se han realizado en la versión v2 b05, aunque en el momento que se escribió este documento está ya disponible la b07. Como contenedor de datos se ha usado la base de datos incrustada "derby".

Para el componente WAR al igual que para el componente EJB, se ha utilizado NetBeans. El desarrollo de los archivos jsp se ha hecho en el editor de este programa. Para la comunicación con el modelo, como se vio en el diseño, se han implementado una serie de clases que usando la inyección de recursos acceden a componente encargado de la lógica de negocio. Para la comunicación entre ambos el diseño planteaba unas serie de objetos VO que proponían una solución para disminuir el flujo de datos entre ambos componentes. El resultado final no fue del todo lo esperando, la estructura demasiado compleja de estos VO hace algo difícil mantener su estado de forma correcta. También se implementaron validadotes y acciones que permiten ser llamados desde los componentes de Faces.

Respecto el segundo componente, los mayores problemas surgieron a la hora de realizar la persistencia. Aunque aparentemente sencillo, la creación de entidades debe de esta cuidadosamente diseñada. Al estar todas las clases relacionadas y en la mayoría bidireccionales, actitudes como persistencia en cascada o borrado en cascada pueden provocar excepciones difíciles de evitar. Las transacciones también son parte importante. Para el desarrollo de los controladores de estas entidades se utilizaron EJB sin estado, que permiten obtener y modificar de forma segura estas entidades y su persistencia. Por ultimo una fachada del modulo de negocio que permite el acceso remoto con el componente WAR.

Para poder desplegar la aplicación final hará falta un servidor GlassFish v2 con una versión igual o superior a la beta 5. El JDK de Java también deberá ser una de las últimas versiones, aunque es mejor mirar la documentación de glassfish para ver que versión será mejor instalar. Para crear las bases de datos se a utilizado el modulo toplink de oracle incluido con glassfish que permite generar la estructura automáticamente, esta opción esta activada en el descriptor de despliegue, y apunta al recurso jdbc/__default, así que tras instalar glassfish y poner en funcionamiento tanto en "domain" como la "database" debería desplegarse sin problemas. Es importante no tener otras tablas ya que podría coincidir los nombres y no desplegarse de forma correcta.

También he colocado la página en un ordenador y estará activa durante el final del mes de junio del 2006, la dirección será http://80.24.245.154:9600/adminApp , de esta forma se tendrá acceso a la aplicación de forma más sencilla. Si hubiese algún problema le agradecería me avisasen al correo Ignacio@bacamt.com .

El resultado final es una aplicación funcional, pero no he desarrollado todas las funciones reflejadas en los casos de uso. Especialmente a la hora de borrar no he implementado nada en la capa cliente, y en el componente EJB únicamente se pueden borrar usuarios que sus clientes no estén relacionados con noticias o facturas. También ha faltado parte de la implementación referida a controlar de forma correcta relaciones como por ejemplo el número de personas apuntadas a un evento no supera el número de plazas.

La implementación final se podrá ver en el archivo adjunto adminApp.ear.

3.5 Conclusiones

Java Enterprise Edition es una herramienta impresionante para el desarrollo de aplicaciones distribuidas. Sus EJB, especialmente los sin estado, parecen estar basados en el patrón "peso-pluma", plantean solución ideal para el procesado de consultas, común en las aplicaciones empresariales distribuidas. Las interfaces locales y remotas de estos componentes los hacen aun más potentes y unidos con la inyección y los descriptores de despliegues o las anotaciones, lo convierten en una herramienta fundamental para estas aplicaciones.

El API de persistencia, Java DataBase Conectivity, Java Naming and Directory Interface hacen posible la comunicación con bases de datos de forma eficiente y segura. Los nombres y directorios permiten organizar aplicaciones de igual forma que los nombres de domino organizan paginas Web, y otras herramientas no abarcadas en el proyecto como SOAP, JavaMail o Message Service abren todas las puertas para convertir a JEE en el centro de una aplicación empresarial.

Por otro lado, toda esta potencia y organización de información propuesta por java debe ser entendida para poder ser usada. En el proyecto se ha intentado en la medida de lo posible, pero aunque la aplicación final funciona,

no utiliza la potencia que brinda JEE. La utilización correcta de estos componentes se consigue tras experimentar con ellos, y desarrollar diferentes aplicaciones, por tanto este proyecto cumple con el objetivo del desarrollo de una aplicación, de la practica de las diferentes etapas de un proyecto de software, pero respecto al conocimiento de la plataforma JEE, únicamente forma la base de sus herramientas. Estas herramientas vistas "por encima" se utilizan en la aplicación pero no de la forma totalmente correcta y eficiente, para esto solo la experiencia en otros proyectos y el constante interés por mejorar los diseños para aprovechar mejor estas tecnologías lograran un conocimiento profundo de JEE.

4 Bibliografía

Grama, Helm, Jhonson, Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*[Addison-Wesley 1995]

Floyd Marsinescu. *EJB Design Patterns* [The Middelware Company 2002, Jhon Whiley & Sons, Inc.]

Derek C. Ashmore. *The J2EE Architect's Handbook* [Dereck 2004, DVT Press]

Ed Romman, Rimma Patel, Gerald Brose. *Mastering Entreprise JavaBeans, Third Edition.* [Wiley Publishing, Inc. 2005]

Jayson Falkner, Kevin Jones. *Servlets and JavaServer Pages: The J2EE Technology Web Tier* [Addison-Wesley 2003]

Jim Keogh. *Manual de Referencia: J2EE* [McGraw Hill 2003]

También han sido utilizados gran cantidad de documentos de Sun Microsystems (www.sun.com), con especial interés en java Blueprints. Destacar el tutorial de la versión 5 de JEE (http://java.sun.com/javaee/5/docs/tutorial/doc/index.html).

La comunidad JCP (http://jcp.org).

La pagina del libro *Core J2EE Patterns: Best Practices and Design Strategies* (http://www.corej2eepatterns.com/Patterns2ndEd/index.htm#).